VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ NA STOLE

TABLETOP OBJECT DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN TIMKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19691/2016/xtimko00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Timko Martin**
Obor: Informační technologie
Téma: **Detekce objektů na stole**
Tabletop Object Detection
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu a problematiku detekce objektů v obraze, případně v hloubkových datech (MS Kinect).
2. Seznamte se s platformou ROS (Robotic Operation System) a jejími možnostmi v oblasti zpracování dat ze senzorů a detekce objektů.
3. Vyberte vhodné metody a nástroje a navrhněte modul pro detekci objektů položených na stole.
4. Proveďte experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
5. Dosažené výsledky zhodnoťte a navrhněte možnosti dalšího vývoje.
6. Vytvořte stručný plakát nebo video prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kapinus Michal, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cieľom tejto práce je navrhnúť a vytvoriť modul pre robotickú platformu PR2. Tento modul má detegovať objekty položené pred robotom na stole a následne umožniť prácu s týmito objektmi. Táto práca popisuje metódy detekcie objektov, ktoré boli využité pri implementácii modulu. Ďalej popisuje samotný návrh a implementáciu modulu. Na záver je spomenuté testovanie modulu a zhodnotenie výsledkov.

Abstract

The aim of this thesis is to design and create a module for robotic platform PR2. This module has to detect objects in front of the robot on a table and enable various operations with these objects. This thesis describes methods of object detection which were used in the implementation of the module. The thesis also describes the methods used for design and implementation of this module. The module testing and evaluation of results are mentioned at the end of the thesis.

Kľúčové slová

ROS, Detekcia, Detekcia objektov, Linemod, ViSP, Object Recognition Kitchen, ORK, Point Cloud, PCL, Kinect

Keywords

ROS, Detection, Object detection, Linemod, ViSP, Object Recognition Kitchen, ORK, Point Cloud, PCL, Kinect

Citácia

TIMKO, Martin. *Detekce objektů na stole*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kapinus Michal.

Detekce objektů na stole

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Kapinusa. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Martin Timko
26. apríla 2017

Poďakovanie

Rád by som sa poďakoval pánovi Ing. Michalovi Kapinusovi za vedenie tejto bakalárskej práce, odbornú pomoc a cenné rady pri spracovaní tejto práce.

Obsah

Kapitola 1

Úvod

V tejto dobe sa pomalým, ale istým tempom roboty stávajú stále viac a viac súčasťou nášho každodenného života. Využitie nachádzajú v mnohých sférach života. Aby robot vedel pracovať vo svojom okolí, je nutné, aby vedel detegovať a rozpoznávať jednotlivé objekty, s ktorými môže interagovať.

Cieľom tejto práce je vytvoriť modul pre školského robota PR2, ktorý pre rozoznanie objektov umiestených na stole pred ním a následnú prácu s nimi, aktuálne vyžaduje pomocné AR kódy prilepené na konkrétnych objektoch. Zmyslom práce je túto závislosť na AR kódoch odstrániť, a tým dosiahnuť využitie robota v prostredí prirodzenejšieho skôr pre človeka ako pre robota. PR2 využíva na svoju prácu kamery Kinect. V tejto práci sa teda zameriam na rôzne techniky detekcie objektov, experimentovanie s nimi a na výber vhodných techník pre robota PR2, pre ktorého je implementovaný modul určený.

V nasledujúcich kapitolách popisujem najprv základné pojmy využité v tejto práci, následne návrh riešenia, popis implementácie a v závere uvádzam testovanie modulu na školskom robotovi.

Kapitola 2

Teoretická časť

V tejto kapitole budú vysvetlené základné pojmy a princípy potrebných technológií, s ktorými som sa stretol pri vypracovaní tejto práce ako robotický operačný systém, robot PR2 či Kinect. Rovnako táto kapitola obsahuje popis možných spôsobov spracovania 2D/3D obrazu využitých k detekcii objektov.

2.1 Robotický operačný systém

Robotický operačný systém[1] (ROS) je open source flexibilný framework, ktorý poskytuje rôzne služby pre prácu a vývoj na rôznych robotických platformách. Zahŕňa rozličné nástroje, knižnice, balíčky a konvencie, ktorých cieľom je zjednodušiť vytváranie komplexného správania pre robota. Pôvodnými autormi ROSu boli Willow Garage a inštitút Stanfordovej univerzity Stanford Artificial Intelligence Laboratory. [20] V dobe tvorby a písania tejto práce bola využívaná na robotovi PR2 distribúcia Indigo Igloo určená pre operačný systém Ubuntu 14.04 LTS.

ROS je založený na princípoch komunikácie *uzlov* pomocou *správ*. Pre implementáciu je možné využiť jazyky C++, Python či dokonca aj Lisp. [20] Ak nie je uvedené inak, informácie som čerpal z knihy *Learning ROS for Robotics Programming*. [19]

Balíček

Software v ROSe je organizovaný pomocou balíčkov. Balíčky sú zložky, ktoré obsahujú uzly, knižnice či konfiguračné súbory s popismi všetkých závislostí. Nájdeme tam aj súbory popisujúce ako sa majú dané zdrojové kódy preložiť a aké spustiteľné súbory následne z nich vzniknú a pod.

Uzol

Uzly sú spustiteľné súbory, ktoré vykonávajú konkrétne výpočty a akcie výslednej aplikácie. Uzly sa spolu kombinujú a spolu so sebou vzájomne komunikujú pomocou zasielania správ. Systém ovládania robota sa napríklad skladá z uzlov na ovládanie koliesok, ďalší uzol sa stará o lokalizáciu, ďalší o plánovanie cesty a pod. Aby sa uzly, ktoré chcú spolu medzi sebou komunikovať našli, je potreba hlavného uzlu. Tento uzol sa stará o komunikáciu a taktiež má prehľad o všetkých uzloch bežiacich v systéme. Hlavný uzol je potrebné spustiť ako

[1] http://www.ros.org/about-ros/

prvý uzol v celom ROSe. Jednotlivé uzly môžu dáta vo forme správ vysielať a zároveň aj prijímať.

Správy

Správy umožňujú komunikáciu medzi komponentmi (uzlami) ROSu. Sú to jednoduché dátové štruktúry zahrňujúce štandardné primitívne dátové typy ako napríklad celé čísla, boolean a iné. Rovnako podporujú zanorené štruktúry či polia.

Mechanizmus zasielania správ, cez ktorý si uzly vymieňajú dáta, funguje ako zoskupenie pomenovaných virtuálnych zberníc, cez ktoré sa jednotlivé správy posielajú. Tieto zbernice fungujú ako rúry, do ktorých sa z jednej strany správy vkladajú a z druhej strany sa vyberajú. Pri práci s ROSom sa často môžeme stretnúť s anglickým výrazom *topic*, ktorý označuje práve tieto zbernice.

Pre priamu komunikáciu typu otázka-odpoveď je v ROSe možné využiť služby. Po zavolaní služby klient dostane odpoveď.

Rviz

Pri práci je často potrebné si dáta vizualizovať. Rviz je užitočný nástroj na zobrazovanie priestorových dát získaných napríklad z point cloud dát Kinectu. Dá sa v ňom zvoliť, ktoré správy chceme odoberať, a tým nám Rviz tieto dáta následne vizualizuje. Rovnako umožňuje premietať výstup zo senzorov robota PR2. V ROSe majú jednotlivé senzory a kamery vyhradené a pomenované svoje vlastné rúry, takže práca s nimi alebo samotné zobrazenie v Rviz je rýchle a jednoduché.

Bag súbory

Pre uľahčenie práce a zvýšenie bezpečnosti ponúka ROS možnosť vyvíjať nástroje či aplikácie aj bez nutnosti priameho pripojenia jednotlivých senzorov či kamier. Umožňujú to *bag* súbory. Tieto súbory slúžia pre ukladanie správ. *Bag* súbory sú typicky vytvorené nástrojmi ako `rosbag`, ktorý umožňuje nahrávať ROS správy a ukladať ich do súboru s časovým razítkom. Tieto dáta je možné kedykoľvek prehrať a experimentovať so softwareom bez nutnosti neustále spúšťať robota.

2.2 Robot PR2

Modul, ktorý má byť výstupom tejto práce, má slúžiť pre tohto robota. Ide o otvorenú a mobilnú platformu vyvinutú spoločnosťou Willow Garage, ktorého software plne využíva ROS. Teda všetky funkcie PR2[2] (Personal Robot 2) sú dostupné cez ROS. PR2 bol navrhnutý a vyvinutý tak, aby umožňoval ľahký vývoj a testovanie robotických aplikácií a technológií. [15]

Hardware

Na základni robota tvoria hardware dva Quad-Core i7 Xeon procesory, 24 GB pamäte RAM. Externý a interný hard disk spolu vytvárajú 2 TB pamäte. O napájanie sa starajú nabíjateľné batérie s kapacitou 1,3 kWh typu Lithium-ion. [5]

[2] http://wiki.ros.org/Robots/PR2

Robot sa vie v prípade potreby pohybovať pomocou štyroch koliesok typu Caster a to maximálnou rýchlosťou 1 m/s. Zo základne sa tiahne torzo robota. Minimálna výška je 1330 mm, maximálna až 1645 mm. Ramená robota sú pripevnené k samotnému torzu. Jedno rameno umožňuje do dlane uchytiť a zdvihnúť náklad o hmotnosti až 1,8 kg. Tento náklad vie PR2 presúvať a otáčať podľa potreby. Dlaň robota je schopná rotácie o 130°. Hlava umiestnená na torze obsahuje dve stereo kamery a jeden Kinect. Hlavou je možné otáčať aj zakláňať pre optimálny uhol pohľadu na objekt záujmu. Komunikáciu umožňuje okrem WiFi a Bluetooth aj Gigabitový Ethernetový štandard. [5]

Obr. 2.1: Robot PR2.[3]

2.3 Point Cloud Library

Point cloud je množina dátových bodov v danom súradnicovom systéme. V trojdimenzionálnom súradnicovom systéme sú tieto body zvyčajne definované pomocou X, Y a Z súradníc. Ak sa pridá informácia aj o farbe bodu, tak sa mračno bodov stáva štvordimenzionálnym. Point cloud alebo mračná bodov sa dajú získať pomocou hĺbkových kamier, ako Asus Xtion PRO alebo Kinect V1, kde sa kombinujú infračervené kamery a štandardná RGB kamera alebo sa dá mračno bodov získať s použitím stereo kamier, kde sa počíta rozdiel obrazu medzi jednotlivými kamerami. Ďalšou možnosťou je využiť kamery využívajúce technológiu ToF (Time-of-Flight), ktoré využívajú rýchlosť svetla na meranie vzdialeností medzi kamerou a snímaným objektom. [22]

Point Cloud Library[4] (skrátene PCL) je open source knižnica algoritmov pre spracovanie 2D/3D obrazu a mračná bodov. Celý PCL framework obsahuje množstvo algoritmov najvyššej úrovne pre filtráciu, detekciu objektov, segmentáciu či rekonštrukciu povrchu. Keďže sa jedná o open source software pod BSD licenciou, je možné ho využiť úplne zdarma aj komerčne, rovnako aj na akýkoľvek výskum. PCL je napísaný v C++ a keďže sa jedná

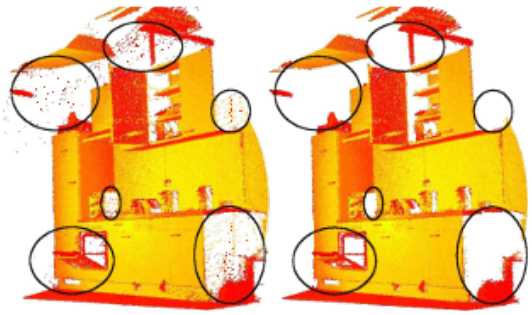
[3] Prevzaté z http://www.willowgarage.com/blog/2010/01/15/pr2-beta-program-call-proposals-out
[4] http://pointclouds.org/about/

o multiplatformový software, je možné s ním pracovať ako na Linuxe, tak aj na Windows, MacOS, iOS a Androide. [22]

Využitie PCL

Kvôli zjednodušeniu vývoja a hlavne používania sa PCL skladá z mnohých menších častí knižníc a modulov, ktoré sú preložiteľné osobitne. Táto modularita je dôležitá pre využitie a distribúciu PCL na rôznych platformách. Možnosti využitia PCL sú teda obrovské a v mnohých oblastiach spracovania obrazu skutočne nájdeme využité PCL mechanizmy. Nasledujúce informácie som získal z webovej stránky dokumentácie PCL. [2]

Jednou z najpoužívanejších knižníc je `pcl_filters`. Táto knižnica obsahuje mechanizmy pre odstránenie nameranej extrémne vybočujúcej hodnoty v dátach alebo na odstránenie šumu pre aplikácie, ktoré sa touto problematikou zaoberajú a využívajú 3D mračná bodov. Pomocou štatistickej analýzy každého bodu a jeho susedných bodov sa dajú odstrániť chybne namerané hodnoty z dát v prípade, že nespĺňajú určité kritéria. Pre orezanie vstupných dát je možné využiť PassThrough filter, ktorý je schopný orezať vstup dát na jednotlivých osiach (x, y, z) podľa obmedzení, ktoré zadá používateľ.



Obr. 2.2: Ukážka odstránenia šumu s použitím PCL filtra. Vzniknuté chyby pri meraní (zakrúžkované oblasti) boli odstránené.[5]

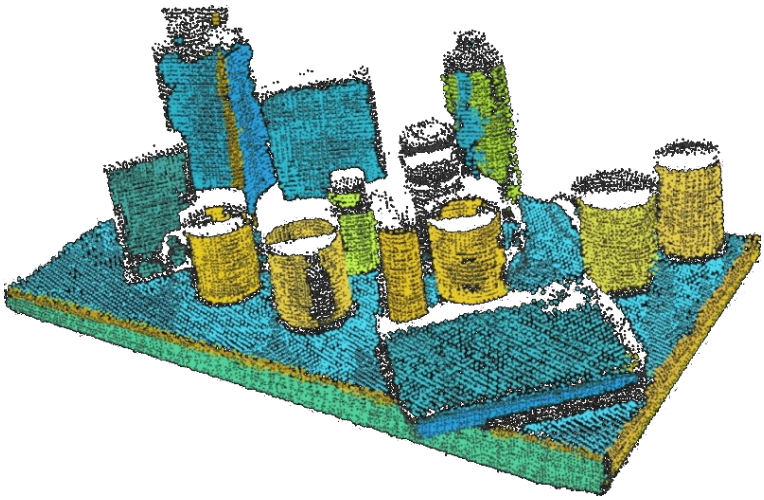
Medzi ďalšie často využívané knižnice sa dá zaradiť aj `pcl_io`, ktorá sa stará o čítanie a vytváranie point cloud súborov, rovnako aj samotné zachytávanie point cloud z rôznych snímacích zariadení ako Kinect, Asus Xtion PRO a i. Pre získavanie dát z týchto zariadení, využíva PCL ovládač OpenNI, ktorý bude popísaný v sekcii 2.7.

Modul `pcl_surface` je schopný si poradiť s rekonštrukciou povrchu z 3D skenov objektov. V závislosti na úlohe môže byť povrch reprezentovaný ako 3D model alebo vyhladený a prevzorkovaný povrch s normálami.

V prípade potreby, PCL ponúka aj dva algoritmy na detekciu keypoint bodov v mračne bodov. Keypoint alebo záujmové body sú body v obraze alebo v mračne bodov, ktoré sú stabilné, výrazné a dajú sa identifikovať pomocou dobre definovaných detekčných kritérií. Dôvodom prečo sú tieto body záujmu špeciálne je, že akokoľvek sa obraz otočí alebo zmení svoju veľkosť, malo by byť možné nájsť rovnaké body záujmu. Knižnica `pcl_keypoints` umožňuje využitie takýchto algoritmov.

[5] Prevzaté z http://docs.pointclouds.org/trunk/group__filters.html

PCL ponúka aj modul `pcl_recognition`, ktorý predkladá možnosť pre aplikácie zaoberajúce sa rozoznávaním objektov, využiť viaceré užitočné algoritmy. Metódy detekcie rovných plôch alebo základých geometrických tvarov nájdeme v `pcl_sample_consensus` knižnici. Pomocou tejto knižnice je možné detegovať bežné rovné plochy ako steny, podlahy a stoly. Rovnako umožňuje detegovať objekty, ktoré majú geometrické tvary ako valec, guľa, kváder (napríklad pohár, plechovka, kniha atď.).



Obr. 2.3: Ukážka detekcie objektov na stole pomocou knižnice PCL.[6]

2.4 OpenCV

OpenCV[7] je open source knižnica zaoberajúca sa počítačovým videním. Celá knižnica je primárne písaná v C a C++. Pracovať s touto knižnicou je umožnené aj v jazykoch ako Python či Java. OpenCV je ako aj PCL cross platforma. To znamená, že nie je závislý len na jednej platforme, ale je dostupný na viaceré platformy ako Linux, Windows, MacOS a rovnako aj iOS a Android. Taktiež zdieľa vlastnosť s PCL o modularite. Niektoré knižnice sú pre viaceré moduly spoločné, naproti tomu niektoré nie. [14]

Vývoj sa zameriaval na výpočtovú efektivitu so silným zameraním na real-time aplikácie. OpenCV vie plne využiť výhody viacjadrových procesorov. Jedným z cieľov, je poskytnúť na použitie jednoduchú infraštruktúru, ktorá pomáha vytvoriť pomerne sofistikované aplikácie dostatočne rýchlo. Knižnica OpenCV obsahuje stovky funkcií, ktoré pokrývajú mnohé oblasti počítačového videnia. Zaoberá sa napríklad videním v robotike, kalibráciou kamier, stereo videním, ďalej aj používateľskými prostrediami a využitie nachádza aj v medicíne či bezpečnosti ako takej. [14]

Vzhľadom na to, že počítačové videnie často súvisí so strojovým učením, OpenCV obsahuje knižnicu Machine Learning Library. Táto knižnica sa stará o štatistické rozpoznávanie vzorov či zhlukovanie. [14]

[6] Prevzaté z http://docs.pointclouds.org/trunk/group__sample__consensus.html
[7] http://opencv.org/about.html

Využitie OpenCV

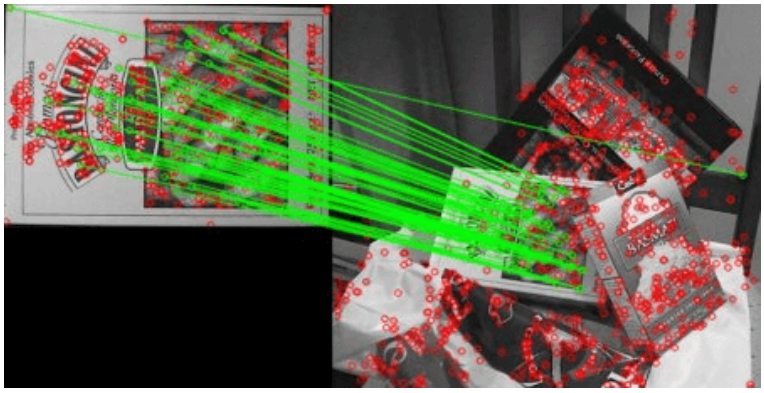
Modul `features2d` poskytuje viaceré možnosti pri detekcii objektov. Predovšetkým k tomu využíva takzvané príznaky (features). Príznaky predstavujú isté jedinečné vlastnosti. V obraze hľadáme konkrétne vzory alebo špecifické vlastnosti, ktoré sú jedinečné a následne je ich možné dostatočne ľahko sledovať a porovnávať. Väčinou sú to rohy a kúty nájdené v obraze. Čiže sa jedná o malé kúsky dát, ktoré sú užitočné pre výpočet podobností medzi obrazmi, ktoré obsahujú dve zložky nazývané feature keypoint a feature descriptor. Keypoint alebo bod záujmu, ktorý napríklad obsahuje 2D pozíciu alebo natočenie a pod. som popísal v predchádzajúcej sekcii 2.3. Deskriptor obsahuje vizuálny popis. Používa sa na porovnávanie podobností medzi jednotlivými obrazovými príznakmi, pomocou ktorých je možné detegovať objekty. [21]

SIFT a SURF

SIFT (skratka zo Scale-invariant feature transform) je algoritmus v počítačovom videní, ktorý sa používa na detekciu a popis príznakov v obraze. Z akéhokoľvek objektu v obraze je možné extrahovať príznakové popisy. Pomocou vzorového obrazu vieme získať jednotlivé príznaky objektu a na základe nich je možné identifikovať objekt, ktorý sa snažíme nájsť napríklad spomedzi viacerých objektov v obraze. SIFT je schopný rozoznať aj objekt, ktorý je čiastočne prekrytý inými objektmi, zmení svoju veľkosť, orientáciu aj pri zmene osvetlenia objektu. [21]

SIFT príznaky sú tvorené výpočtom gradientu v každom pixely okolia o veľkosti 16x16 okolo detegovaného bodu záujmu s použitím primeranej úrovne Gaussovej pyramídy. Obrazové pyramídy, medzi ktoré patrí aj Gaussová pyramída sú súbory rovnakých obrazov s rôznym rozlíšením. Tieto obrazy sú tvorené z originálneho obrazu, a to opakovaným zmenšovaním a vyhladzovaním. Týmto spôsobom vie SIFT detegovať objekt z obrazu aj v prípade väčšej vzdialenosti od kamery, ako bol pôvodne vzorový objekt. [21]

Zrýchlenú verziu algoritmu SIFT môžeme nazvať SURF (Speeded Up Robust Features). SURF je založený na rovnakých princípoch a krokoch ako SIFT, ale po úpravách jednotlivých krokov je vo výsledku rýchlejší. V OpenCV nájdeme využitie oboch týchto algoritmov.

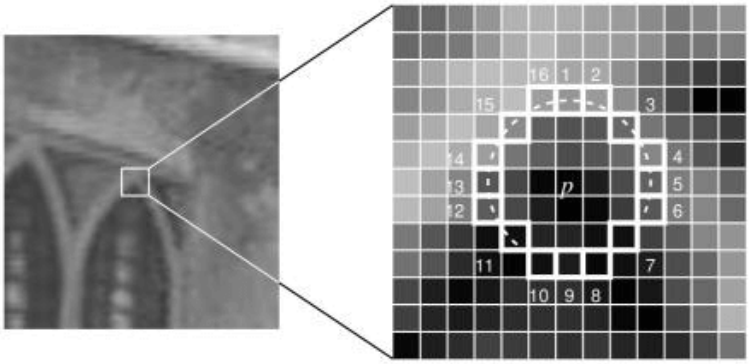


Obr. 2.4: Ukážka rozoznania objektu v zhluku iných objektov pomocou algoritmu SIFT.[8]

[8] Prevzaté z http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

FAST a ORB

Features from Accelerated Segment Test alebo skrátene FAST je metóda detekcie rohov, ktorá sa využíva pre extrakciu príznakov z obrazov. Vývoj bol zameraný na výpočtovú efektivitu. Oproti SIFT nevyužíva štvorcové okolie, ale kružnicu na určenie, či sa jedná o roh. Stredom kružnice s polomerom 3 pixely je potencionálny bod záujmu a obvod kružnice tvorí 16 pixelov, kde sa sleduje intenzita jasu jednotlivých častí kružnice. Na základe podmienok sa rozhoduje, či sa jedná alebo nejedná o roh. [18]



Obr. 2.5: Ukážka detekcie rohu v obraze pomocou algoritmu FAST.[9]

Metódy SIFT a SURF sú patentované, to znamená že ich použitie nie je úplne zdarma. Preto OpenCV vývojári prišli s riešením v podobe algoritmu ORB (Oriented FAST and Rotated BRIEF). Jedná sa o efektívnu alternatívu k SIFT a SURF ale tentokrát využitie nie je spoplatňované. ORB vznikol zlúčením algoritmu FAST, ktorý najprv vyhľadá body záujmu a využitím algoritmu BRIEF, ktorý následne vykonáva funkciu deskriptora. ORB prišlo s mnohými modifikáciami pre celkové vylepšenie výkonu. Jednou z nevýhod FAST je, že nevyhodnocuje orientáciu bodov záujmu. ORB rieši aj tento problém a zaručuje rýchle a efektívne detegovanie obrazových príznakov. [8]

Porovnávanie príznakov

V predchádzajúcich sekciách som popísal niektoré algoritmy, ktoré sa využívajú v OpenCV pri detekcii objektov. Tieto techniky vedia rozpoznať jednotlivé body záujmu, no samotné porovnávanie dvoch obrazov a porovnávanie príznakov prenechávajú na iné algoritmy.

Prvým je Brute-Force Matcher, ktorý vezme deskriptora jedného príznaku z prvého obrazu a porovná všetky ostatné príznaky z druhého obrazu. Najpodobnejší príznak vyberie a označí za zhodný v porovnávaných obrazoch. Celou knižnicou, ktorá umožňuje porovnávanie je FLANN (Fast Library for Approximate Nearest Neighbors), ktorá je rýchlejšia vo veľkých datasetoch oproti metóde Brute-Force. FLANN obsahuje algoritmy optimalizované pre hľadanie týchto zhôd. Vie vybrať vhodné algoritmy pre rôzne veľké a zložité datasety. V OpenCV je FLANN jedným z modulov a je schopný pracovať okrem 2D aj s 3D dátami a preto ho nájdeme aj v PCL. [3]

[9] Prevzaté z http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html

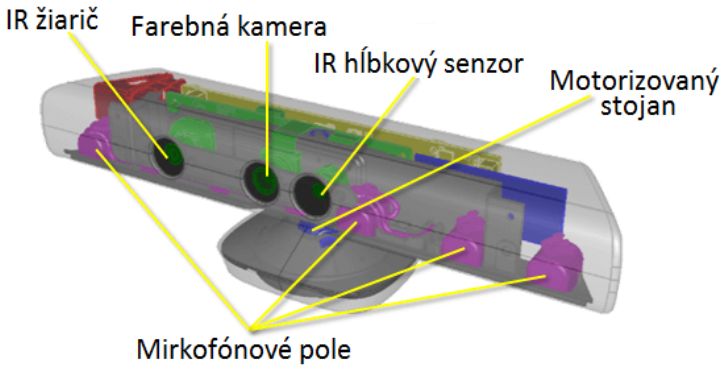
2.5 Kinect V1

Kinect bol pôvodne vyvíjaný spoločnosťou Microsoft pre hernú konzolu Xbox 360. Jedná sa o zariadenie, ktoré umožňuje snímať pohyb, a tým ovládať samotnú konzolu. Hráč nepotrebuje žiaden ovládač, keďže ovládačom sa stáva samotný hráč. Vydanie Kinectu v roku 2010 spustilo revolúciu v hernom svete a po krátkom čase sa Kinect prestal limitovať len na hernú konzolu. Vývojári po celom svete začali toto zariadenie používať pre vývoj svojich aplikácií. Uplatnenie našiel aj v robotike, kde poskytuje pomerne lacnú a kvalitnú kameru, ktorá často slúži ako oči a uši robota. Celú túto sekciu o prvej verzii Kinectu som čerpal z knihy od autora Abhijit Jana. [16]

Komponenty

Po vydaní Kinectu na Xbox 360 bol vydaný aj Kinect upravený pre prácu s PC nazvaný ako Kinect for Windows, ktorý odštartoval veľký záujem o software spojený so snímaním pohybu či okrem iného, použitím Kinectu pre detekciu pohybu alebo objektov. Jednou z úprav je pridanie nového firmware, ktorý umožňuje sledovanie objektov od kamery vzdialených už od 40 cm. Verzia pre Xbox 360 umožňovala najbližšie snímanie až od vzdialenosti 80 cm. Celé zariadenie sa skladá z viacerých komponentov. Hlavnými dôležitými komponentmi sú:

- farebná RGB kamera,
- infračervený (IR) žiarič,
- infračervený (IR) hĺbkový senzor,
- motorizovaný stojan,
- mikrofónové pole.



Obr. 2.6: Základné komponenty Kinect V1.[10]

Ďalej, okrem už spomenutých komponent, Kinect pre prepojenie s PC potrebuje napájací adaptér a USB adaptér.

[10] Prevzaté z https://msdn.microsoft.com/en-us/library/jj131033.aspx a upravené v GIMP

Farebná RGB kamera, ktorá je zodpovedná za snímanie farebného obrazu, umožňuje rýchlosť snímania 30 snímkov za sekundu pri rozlíšení 640x480, prípadne až 1280x960 pixelov pri rýchlosti 12 snímkov za sekundu. Veľkosti pozorovacích uhlov sú 43° vertikálne a 57° horizontálne.

Infračervený žiarič neustále vyžaruje infračervené svetlo, vo forme jednotlivých bodov, ktoré sa rôzne odrážajú na rôznych objektoch. Následne tieto body využije infračervený hĺbkový senzor, ktorý zmeria jednotlivé vzdialenosti medzi bodmi a senzorom, a tým tvorí hĺbkové snímanie. Hĺbkové dáta je možné snímať v rozlíšení 320x240 a 80x60 pixelov. O spracovanie dát a vytváranie samotného hĺbkového obrazu sa stará PrimeSense čip, ktorý riadi infračervený žiarič aj senzor.

Mikrofónové pole tvoria štyri odlišné mikrofóny, ktoré umožňujú nie len zachytiť zvuk, ale aj určiť smer zvukovej vlny. Výhodou využitia mikrofónového poľa je aj efektívnosť zachytenia zvuku a odstránenie ozveny pričom sa potlačuje nechcený šum.

Motorček zabudovaný v stojane umožňuje zmeniť uhol kamery pre optimálne snímanie. Je schopný zmeniť pozorovací uhol vertikálne až o 27°. LED indikátor umiestnený medzi farebnou kamerou a infračerveným žiaričom po zapojení do PC indikuje zeleným svetlom, že všetky ovládače sa načítali korektne a Kinect je plne pripravený na používanie.

Využitie Kinect V1

Kinect je možné využiť v mnohých aspektoch života ako v zdravotníctve, školstve, je možné ho využiť ako osobného trénera ale aj v robotike. Ako som už spomínal, Kinect často slúži ako oči a uši rôznych robotov. Okrem zachytávania real-time videa umožňuje sledovať ľudské telo a reagovať na pohyby, gestá a vytvoriť tak bezdotykové užívateľské rozhranie. Pomocou Kinectu vieme merať vzdialenosti objektov umiestnených od 40 cm až po 4 m od senzorov. Kinect nám umožňuje analyzovať 3D dáta a vytvárať 3D modely, rovnako aj vytvárať hĺbkové mapy sledovaných objektov. Umožňuje tak robotom lepšie sledovať svoje okolie ako pomocou obyčajnej digitálnej kamery. V neposlednom rade poskytuje možnosti pomocou mikrofónov vytvárať aplikácie pre rozoznávanie hlasu človeka a ovládanie aplikácií hlasom.

Obr. 2.7: Zariadenie Kinect V1 (vľavo), zobrazenie hĺbkového obrazu stola s na ňom položenými objektmi, vytvoreného pomocou Kinect V1 (vpravo).[11]

[11] Prevzaté z https://www.ifixit.com/Teardown/Xbox+360+Kinect+Teardown/4066 a http://www.telecom.ulg.ac.be/publi/publications/lejeune/Lejeune2011ANewJump/index.html

2.6 Kinect V2

Druhá generácia Kinect kamier bola vydaná koncom roka 2013 spoločne s konzolou Xbox ONE. Rovnako ako prvý Kinect, aj tento má za úlohu zjednodušiť interakciu konzoly s hráčom. Taktiež ako pri prvej verzii je umožnené vývojárom vlastných aplikácii pracovať s touto vylepšenou verziou Kinectu. Celú túto sekciu o druhej verzii Kinectu som čerpal z bakalárskej práce študenta univerzity z Tartu Lembita Valgma. [23]

Komponenty

Kinect V2 slúži ako ToF (Time-of-Flight) kamera. Na základe známej rýchlosti svetla vie zmerať čas letu svetelného signálu emitovaného lasermi medzi kamerou a svojím okolím, čím získava vzdialenosti jednotlivých bodov obrazu od kamery. Na tomto princípe je založená práca hĺbkového senzora umiestneného na kamere Kinect. Hlavné komponenty sú:

- farebná RGB kamera,
- 3D hĺbkový senzor s lasermi,
- mikrofónové pole.

Obr. 2.8: Základné komponenty Kinect V2.[12]

Farebná RGB kamera je schopná snímať obraz s rozlíšením 1920x1080 pixelov pri rýchlosti až 60 snímkov za sekundu. Spomínaný hĺbkový senzor pracuje pri rozlíšení 512x424 pixelov. S vyšším rozlíšením je umožnené sledovať väčšie detaily. Napríklad kým s prvým Kinectom bolo možné sledovať pozíciu ruky v priestore, s druhým je možné sledovať okrem pozície aj natočenie ruky či jednotlivé prsty. Kinect V2 umožňuje sledovať objekty vzdialené od kamery minimálne 50 cm a maximálne 8 metrov.

Veľkosti pozorovacích uhlov sa rovnako zvýšili. Vertikálny pozorovací uhol sa zväčšil o 19° na 60°. Horizontálny uhol sa taktiež zväčšil, a to na 70°, čo činí zväčšenie oproti Kinectu V1 o 13°. Mikrofónové pole tvoria naďalej štyri mikrofóny, ktoré umožňujú efektívne zachytiť zvuk a potlačiť šum. Motorizovaný stojan sa pri novšej verzii nenachádza. Namiesto neho Kinect obsahuje manuálne nastaviteľný stojan, s ktorým je možné nastaviť požadovaný uhol kamery. Pre prepojenie zariadenia s PC je potrebný Kinect V2 adaptér.

[12] Prevzaté z https://www.slideshare.net/MatteoValoriani/programming-with-kinect-v2 a upravené v GIMP

Využitie Kinect V2

Kinect V2 je možné využiť v rovnakých sférach ako som už popísal v sekcii 2.5. Keďže ide o vylepšený Kinect, umožňuje ešte lepšie využitie. V oblasti medicíny sa začínajú vďaka dokonalejšiemu senzoru vyvíjať aplikácie, ktoré nájdu využitie v nemocniciach. Dokáže totiž zmerať približný tep človeka a pod. Taktiež vo firemnom prostredí vie slúžiť ako kvalitná Full HD kamera pre video konferencie a vie poslúžiť ako prekladač posunkového jazyka pre sluchovo postihnutých.

Obr. 2.9: Ukážka zobrazenia scény získanej pomocou hĺbkového senzoru Kinectu V2.[13]

2.7 OpenNI a OpenKinect

OpenNI[14] je nezisková organizácia založená s cieľom o zvýšenie interoperability prirodzených užívateľských rozhraní (Natural User Interfaces) pre prirodzenú interakciu so zariadeniami a aplikáciami. OpenNI sa využíva pre uľahčenie prístupu pri využití týchto zariadení, pomocou poskytnutia middleware na prácu s nimi. Napríklad Kinecty popísané v predchádzajúcich sekciách 2.5 a 2.6 spadajú do tejto kategórie zariadení. Jedná sa o open source cross platformu umožňujúcu vývojárom vytváranie aplikácií pre široké spektrum odvetví, medzi ktoré sa nepochybne radí aj robotika. [17]

Na druhej strane OpenKinect[15] tvorí otvorená komunita ľudí, ktorá sa zameriava iba na Kinect a jeho využitie pre vývojárov. Komunita OpenKinect o veľkosti cez 2000 členov vytvorila open source knižnice, ktoré dovoľujú využiť Kinect na Windowse aj na Linuxe a MacOS. Výsledkom práce je software nazvaný libfreenect. [17]

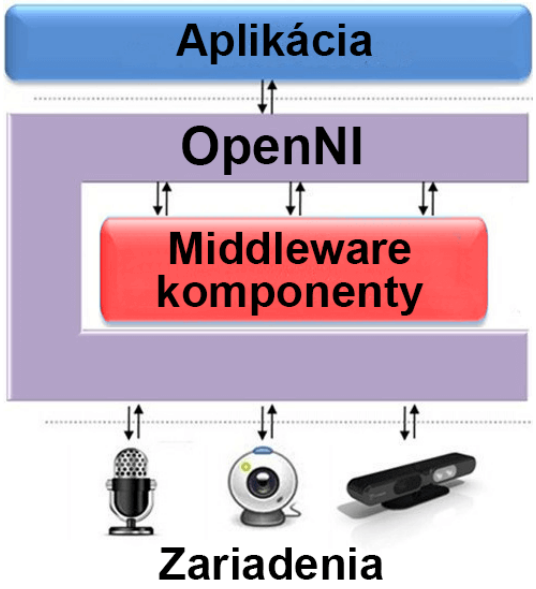
[13] Prevzaté z http://www.gadgetguy.com.au/hands-on-with-the-xbox-one/microsoft-xbox-one-hands-on-2013-16-kinect-depth/

[14] http://openni.ru/about/index.html

[15] https://openkinect.org/wiki/Main_Page

Využitie

Ako OpenNI tak aj OpenKinect poskytujú využitie v ROSe. ROS OpenNI balíček poskytuje ovládače pre prácu s Kinectom. Rovnako poskytuje možnosti využiť balíček pre sledovanie kostry a gest človeka pomocou NiTE[16] middleware. Obe knižnice teda ponúkajú možnosti práce s Kinectom, a to priamo v ROSe. Jedným z rozdielov je, že libfreenect poskytuje možnosť využitia motorizovaného stojana Kinectu V1, ovládanie LED, kým OpenNI ovládač to neumožňuje. Libfreenect sa zameriava na ovládnie aj na nižšej úrovni. Na druhú stranu OpenNI ponúka bohaté možnosti funkcií spracovania hĺbkových a obrazových dát ako prácu s NiTE middleware, ktorý je v tejto chvíli považovaný za najpokrokovejší v oblasti sledovania kostry človeka a detekcie rúk pomocou hĺbkových kamier.



Obr. 2.10: Vizualizácia architektúry OpenNI.[17]

Obe knižnice teda ponúkajú možnosti prístupu k dátam z Kinectu, ktoré sa dajú využiť pri detekcii objektov. Rozdiel tkvie v tom, že OpenNI pracuje so zariadeniami na okrem nižšej úrovne aj na vyššej úrovni ako libfreenect a poskytuje okrem prístupu k dátam aj iné možnosti a funkcie priamo na prácu s týmito dátami.

[16] http://openni.ru/files/nite/

[17] Prevzaté z http://yannickloriot.com/2011/03/kinect-how-to-install-and-use-openni-on-windows-part-1/ a upravené v GIMP

Kapitola 3

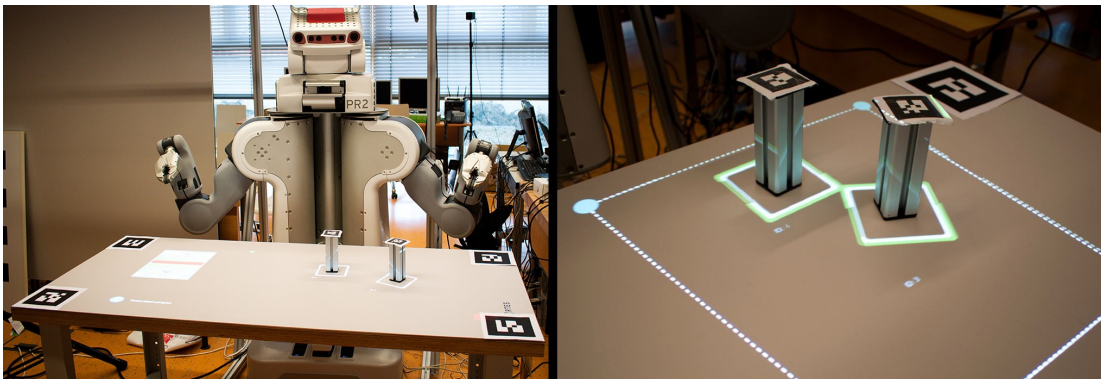
Návrh riešenia

V tejto kapitole popisujem návrh tejto bakalárskej práce. Najprv podrobnejšie špecifikujem problém, následne predstavím niektoré zaujímavé existujúce riešenia problému detekcie objektov. V ďalších sekciách popíšem návrh výberu vhodných techník, s ktorými budem pracovať a návrh architektúry vyvíjaného programu. Na záver spomeniem návrh postupu riešenia.

3.1 Špecifikácia problému

Cieľom tejto práce je vytvorenie modulu pre školského robota PR2. Tento modul vo forme uzlu ROSu bude slúžiť ako akýsi most, ktorý bude pracovať s už existujúcou technikou detekcie objektov a bude ju spájať s robotom PR2.

Pracovisko v školskom robotickom laboratóriu mimo ľudského pracovníka obsahuje aj stôl (pracovne nazvaný ARTable, zobrazený na obrázku 3.1) a robotickú platformu (robot PR2). V tejto chvíli je pre prácu robota s objektmi na ARTable nutné využívať AR kódy vo forme štítkov umiestených na objektoch (obr. 3.1). Tieto AR kódy výrazne obmedzujú akúkoľvek prácu s objektmi položenými na stole. Pomocou môjho uzla by teda nebolo potrebné využívať práve tieto štítky pre interakciu PR2 s objektmi na ARTable. To znamená priblíženie práce robota viac k ľudským podmienkam.



Obr. 3.1: Vľavo ukážka stola ARTable s robotom PR2. Vpravo ukážka položených objektov na ARTable s pripevnenými AR štítkami.

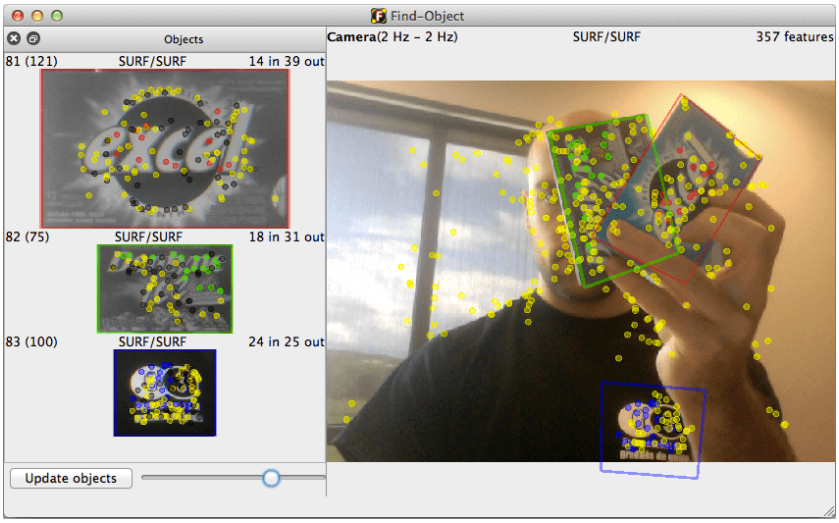
3.2 Existujúce riešenia

Problémom detekcie objektov sa zaoberajú mnohí vývojári. Keďže celá práca je robená pre školského robota PR2, ktorý využíva ROS, tak sa zameriam na existujúce techniky a riešenia bežiace pod ROSom. Samotný ROS ponúka viaceré riešenia, pričom každé má svoje výhody aj nevýhody. Balíček `find_object_2d` stojí za zmienku, ale pre detekciu a rozoznávanie objektov sa ponúka v ROSe využiť Object Recognition Kitchen. Ďalšou alternatívou môže byť napríklad aj platforma ViSP.

Balíček find_object_2d

Tento ROS balíček ponúka pre detekciu okrem iných OpenCV algoritmov aj implementované algoritmy popísané v predchádzajúcej kapitole 2.4 ako SIFT, SURF, FAST, BRIEF a ORB. Balíček je výsledkom integrácie aplikácie Find-Object[1] do ROSu, ktorý ponúka pre uľahčenie práce aj používateľské rozhranie. [4]

Ako už názov naznačuje, pre prácu postačuje akákoľvek 2D kamera. Postačuje dokonca aj webkamera počítača, s ktorou je možné detegovať objekty. Pri použití 3D kamery ako Kinect, je možné okrem štandardných dát, o aký objekt sa jedná v podobe publikovaných ROS správ, zistiť aj 3D pozíciu v sledovanom priestore. [4]



Obr. 3.2: Ukážka rozpoznania objektov pomocou aplikácie Find-Object s využitím detektora a deskriptora príznakov SURF.[2]

Object Recognition Kitchen

Object Recognition Kitchen (skrátene ORK) je projekt, ktorý pôvodne začal vyvíjať Willow Garage a následne pomocou komunity vznikli nakoniec štyri rôzne techniky pre detekciu a následné rozoznanie objektov. Samozrejmosťou pre tieto techniky je aj zistenie 3D pozície objektov na scéne, ktorú sledujeme. V tejto chvíli nie je žiadna unikátna metóda pre rozoznávanie objektov, ktorá by bola schopná rozoznávať textúrované, netextúrované či

[1] http://introlab.github.io/find-object/
[2] Prevzaté z http://wiki.ros.org/find_object_2d

transparentné objekty naraz. ORK sa ale usiluje tomuto ideálu priblížiť a snaží sa zlúčiť viaceré algoritmy a techniky pre detekciu objektov a ich rozoznávanie. Stará sa o rôzne aspekty a rieši aj bežné problémy, ktoré nesúvisia priamo s obrazovým spracovaním ako správa databázy, manipuláciu so vstupmi, výstupmi a integráciu do robotov či priamo do ROSu. ORK taktiež uľahčuje opätovné znovupoužitie kódu. [6]

Techniky frameworku ORK, ktoré boli zatiaľ implementované sú *Tabletop, LINE-MOD, Texture Object Detection* a *Transparent Objects*.

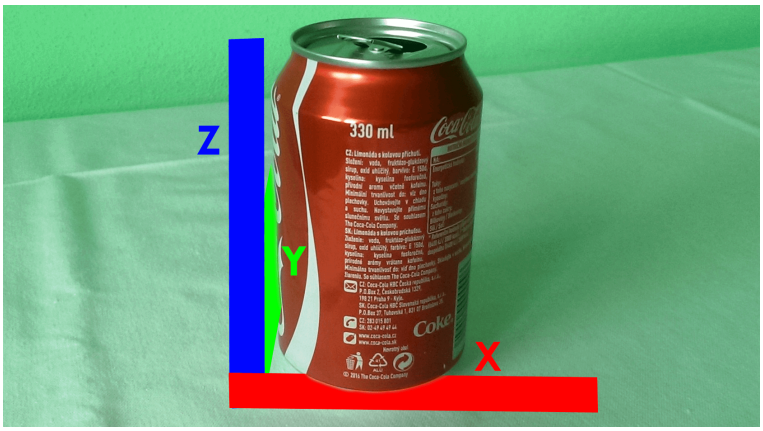
Tabletop

Technika Tabletop, ako je popísaná na dokumentačných stránkach,[10] je portom z pôvodného ROS balíčku `tabletop_object_detector`. Ide o najstaršiu techniku v rámci ORK pre detekciu objektov. Táto metóda sa skladá z dvoch častí. Najprv sa hľadá stôl až následne sa rozoznávajú objekty. Tabletop vyžaduje point cloud dáta zo stereo kamier či Kinectu.

Prvým krokom je segmentácia. Hľadájú sa dominantné rovinné plochy pomocou analýzy 3D normálových vektorov v point cloud dátach. Takáto nájdená plocha sa považuje za stôl. Pomocou viacrozmernej štatistickej metódy, zhlukovej analýzy, sa klasifikujú objekty na stole. Teda zhluky bodov ležiace nad plochou stola sú považované za objekt.

Nasleduje rozoznávanie objektov, kde sa pre každý zhluk iteratívne prejde celá databáza a porovnáva sa podobnosť s 3D modelom v databáze. Ak je zhoda s modelom dostatočná, tak sa tento objekt považuje za rozoznaný, metóda vracia ID modelu z databáze spoločne s informáciou, ktorý zhluk odpovedá danému objektu. Tabletop využíva pre segmentáciu a rozoznávanie funkcie z knižníc PCL a rovnako aj OpenCV.

Tabletop vie rozoznať tuhé, rotačne symetrické objekty, ktoré spĺňajú Lambertov zákon o odraze svetla, čiže ktoré sú dostatočne matné. Priehľadné a lesklé objekty nie je schopný detegovať. Metóda vyžaduje, aby bol objekt položený na stole vzpriamene (os Z smerujúca hore vzhľadom k stolu ako na obrázku 3.3) bez 3D rotácií osí X a Y (tolerancia 2°), pretože pri porovnávaní objektov operuje v 2D. Metóde nezáleží, či je alebo nie je objekt textúrovaný. V prípade textúry objektu ju ignoruje. Textúra nemá žiaden vplyv na detekciu.



Obr. 3.3: Ukážka vzpriameného objektu (os Z smeruje hore vzhľadom k stolu) pripraveného pre detekciu pomocou metódy Tabletop.

Linemod

Metóda Linemod alebo LINE-MOD je považovaná za jednu z najlepších metód pre rozoznávanie generických, tuhých objektov s veľmi rýchlym rozoznaním, o aký objekt sa jedná. Autorom je Stefan Hinterstoisser[3], ktorý k implementácii využil OpenCV. Objekty v databáze je najprv nutné natrénovať, až potom je možné ich detegovať. [7]

Pre natrénovanie objektov z databáze Linemod generuje tisíce pohľadov okolo objektu, pričom využíva knižnicu Random View Generator, ktorá je v ROSe dostupná ako balíček `object_recognition_renderer`. Táto knižnica generuje rôzne pohľady na objekt. Jednotlivé pohľady sú generované na vstupný 3D model objektu podľa daných pravidiel. Na 3D model sa aplikujú zmeny uhlov pohľadu, rotácie a zmeny mierky. Výstupom pre každý objekt je OpenCV n-dimenzionálna matica popisujúca pohľady. V prípade desiatkov objektov v databáze môže trénovanie trvať minúty až desiatky minút. [9]

Linemod ako aj Tabletop nevie detegovať priehľadné a lesklé objekty. Taktiež nepracuje s textúrami objektov. Linemod nevie rozoznať čiastočne prekryté objekty. Pre prácu potrebuje 3D kamery ako Kinect alebo Xtion PRO. Na rozdiel od Tabletop nie je limitovaný pozíciou alebo rotáciou objektu a je schopný detegovať a rozoznať o aký objekt sa jedná pri akomkoľvek uhle pozorovania objektu. [6]

Texture Object Detection

Texture Object Detection (skrátene TOD) je prvou a zároveň zatiaľ jedinou technikou v ORK, ktorá sa zaoberá aj textúrami objektov. Teda okrem 3D modelu objektu, potrebuje aj textúru konkrétneho objektu, ktorého chceme rozoznať. [11]

Pre natrénovanie objektov v databáze je nutné zadať techniku hľadania príznakov a deskriptora, ktorý bude slúžiť pri rozpoznávaní textúr. Využívajú sa OpenCV techniky. Predvoleným nastavením je technika ORB z OpenCV modulu `features2d`. TOD vie pracovať iba s manuálne vytvorenými 3D modelmi objektov. To znamená, že je nutné najprv tieto objekty zachytiť pomocou pomocných techník, ktoré ORK ponúka na výrobu vlastných 3D modelov. [11]

Pomocou neustáleho počítania príznakov z obrazu pri detekcii sa tieto príznaky porovnávajú s textúrami objektov v databáze, a tým je možné vykonať rozoznanie objektov na základe textúry. TOD okrem OpenCV potrebuje aj point cloud pre zistenie konkrétnej pózy objektu. Na to využíva kamery, ktoré umožňujú hĺbkové snímanie. [11]

Transparent Objects

Na otázku čo s priesvitnými, transparentnými objektmi odpovedá technika Transparent objects. Táto technika je schopná detegovať aj priesvitné objekty, ako napríklad sklenené poháre.

Metóda pre natrénovanie objektov potrebuje 3D modely týchto objektov. Trénovací algoritmus postupne rotuje 3D model a ukladá si siluety z každého pohľadu do databáze. Pri rôznych uhloch má objekt rôznu siluetu a následne pri rozoznávaní o ktorý objekt sa jedná, algoritmus porovnáva tieto siluety. Algoritmus zisťuje, ktorá silueta v databáze sa hodí najviac k detegovanej siluete. [1]

Pre hľadanie siluety v obraze sa možní kandidáti zisťujú pomocou hĺbkového obrazu z Kinectu. Priehľadný objekt vytvorí v hĺbkovom obraze *not a number* alebo tzv. NaN región. Následne segmentačná metóda GrabCut vystrihne tento región, ktorý je považovaný

[3] Viac informácií o prácach na http://far.in.tum.de/Main/StefanHinterstoisser

za siluetu. Algoritmus porovnávaním siluet vie zistiť konkrétnu pózu objektu z obrazu, a tým vie zistiť o aký objekt sa jedná. [1]

ViSP

Skratka ViSP[4] vznikla skrátením Visual Servoing Platform. Jedná sa o modulárnu cross platformovú knižnicu, ktorá ponúka techniky pre vývoj aplikácii s využitím vizuálnych sledovaní (visual tracking). ViSP ponúka funkcie pre spracovanie obrazu a sledovanie objektov v reálnom čase. Je preto možné využiť ho v robotike. ViSP je implementovaný v jazyku C++ pomocou open source knižníc ako OpenCV. [12]

ViSP je integrovaný do ROSu vo forme viacerých balíčkov. Pre detekciu a sledovanie obektov je možné využiť dva balíčky. Prvým je `visp_tracker`, ktorý tvorí tzv. wrapper okolo ViSP techník *model-based tracker* a *KLT tracker* a umožňuje ich využitie v ROSe pomocou funkčného uzla. Táto technika potrebude 3D CAD model objektu a je schopná rozoznať objekt a sledovať ho pomocou viditeľných hrán objektu. Vie sledovať textúrované aj netextúrované objekty. Potrebné je ale pri každom spustení objekt inicializovať zadaním presnej polohy daných rohov objektu. [13]

Druhým balíčkom je `visp_auto_tracker`, ktorý pracuje podobne ako `visp_tracker`, ale nepotrebuje zakaždým inicializáciu objektu. Naviac potrebuje informáciu o presnom umiestnení QR alebo flash kódu umiestnenom na konkrétnom objekte. Po načítaní kódu táto metóda automaticky inicializuje objekt v obraze a začne ho sledovať. Ide teda o zjednodušenie a zautomatizovanie inicializácie. [13]

Sledovanie objektov pomocou AR kódov implementované na ARTable v tejto chvíli potrebuje, aby AR kód bol celý čas jasne a zreteľne viditeľný, s minimálnou manipuláciou objektu. Pri `visp_auto_tracker` implementácii slúži kód iba pre inicializáciu polohy objektu a následne sa o sledovanie objektu starajú spomínané techniky *model-based tracker* a *KLT tracker*. To umožňuje väčšiu mieru manipulácie s objektom na stole.

3.3 Návrh architektúry uzlu

Tento uzol by som mohol nazvať ako wrapper okolo metód pre detekciu objektov. Jeho úloha je zabezpečiť správny vstup pre ďalšie uzly pracujúce s ARTable a PR2. To znamená, že môj uzol by mal vybrať z jednotlivých správ, ktoré v ROSe prúdia pri detekcii objektov tie, ktoré sú potrebné pre už implementovaný modul ARTable v školskom robotickom laboratóriu.

K správnemu prepojeniu modulov techník detekcie objektov a modulu ARTable je potrebné dodržať spoločné rozhranie. Tým rozhraním sa rozumie istá štruktúra ROS správ. Môj uzol bude odoberať správy, ktoré publikujú jednotlivé metódy detekcie. Tieto správy spracuje a vyberie potrebné informácie a upraví ich na správy správneho tvaru a typu, ktoré očakáva modul ARTable. Následne tieto správy publikuje pre ARTable na základe dohodnutých konvencií. Týmto spôsobom sa dostanú správne informácie zo zmesi rôznych dát z techník detekcií objektov pomocou môjho uzla až k ARTable a PR2.

3.4 Návrh postupu výberu vhodných techník

Keďže všetky spomenuté techniky detekcie objektov v predchádzajúcej sekcii 3.2 sú bežiace pod ROSom, sú vhodnými kandidátmi pre výber.

[4] http://visp.inria.fr/

Základné podmienky, ktoré by mali metódy detekcie spĺňať, sú dve. Metóda pri rozoznávaní musí jasne určiť, o aký objekt sa jedná a má poskytovať informácie o pozícii, kde presne sa v priestore nachádza. Keďže implementovaný uzol má mať jasne dané výstupy, tieto dve podmienky sú preň kľúčové. Bez týchto informácii, o aký objekt sa jedná a kde sa nachádza, by môj uzol nebol schopný poskytnúť zmysluplný výstup.

Ďalej by mala technika rozpoznať objekty pri akejkoľvek póze. To znamená, že obmedzenia, pri ktorých technika vie rozoznať objekt položený akýmkoľvek spôsobom na stole, by mali byť minimálne. Napríklad pri rôznych rotáciách okolo osí objektu, aby sa to priblížilo čo najviac realite, kde človeku často nezáleží na tom, ako je objekt položený. Takže krabica tvaru kvádra tak môže byť položená na všetkých svojich stranách bez rozdielu.

V bežnom živote sa často stretávame okrem objektov s textúrou aj s objektmi bez textúry (napr. obyčajný hrnček na kávu bez potlače). Práve dôležitosť textúry objektu je otázna časť. Pri dvoch rozdielnych nápojoch (rozdielna textúra objektov) uložených v identických plechovkách (rovnaký tvar oboch objektov) je pre techniky, ktoré neberú textúru objektu do úvahy, nemožné tieto dva objekty rozoznať ako rozdielne. V tejto práci preto budem skôr uprednostňovať detekciu objektu samotného ako rozoznávanie, o akú variantu rovnakého objektu sa jedná. Takže to znamená, že uprednostňujem informáciu, o aký objekt ide (napr. plechovka, hrnček, loptička) pred informáciou, o akú farebnú kombináciu sa jedná (napr. o akého výrobcu nápoja ide alebo o akú značku tenisovej loptičky sa jedná).

Takže pri technikách budem sledovať tieto štyri spomenuté aspekty:

- schopnosť správne určiť o aký objekt sa jedná,
- schopnosť určiť presnú pozíciu rozoznaného objektu,
- schopnosť rozoznať objekt pri čo najmenších obmedzeniach,
- schopnosť rozoznať objekt skôr na základe tvaru, ako na základe textúry, čo ho pokrýva.

Podľa povahy jednotlivých techník vyberiem na základe týchto aspektov jednu až dve, s ktorými následne budem pracovať ďalej a využijem ich aj pri vytváraní uzlov. Pre každú vybranú techniku bude vytvorený osobitný uzol.

3.5 Návrh postupu riešenia

Robotické laboratórium obsahuje všetko potrebné pre prácu vrátane ROSu na počítačoch, hĺbkových kamier, ARTable a samotného robota PR2. V úvodných fázach ale nebude potrebné pracovať priamo v laboratóriu, keďže pre výber vhodných techník a implementáciu uzlov postačuje aj vlastný osobný počítač s Kinectom.

Najprv je nutné zoznámiť sa s ROSom. Následne budem skúmať jednotlivé techniky detekcie pomocou Kinectu a budem vyberať tie najvhodnejšie podľa kritérií v 3.4. Potom sa zameriam na vytvorenie uzlov a po ich vytvorení sa pôjdem presvedčiť o ich kompatibilite s ARTable a PR2 v školskom laboratóriu. Tam sa budú testovať a vyladovať. Predpokladám, že niektoré skutočnosti a problémy sa odhalia až priamo pri práci v laboratóriu. Preto je nutné tieto uzly testovať priamo v laboratóriu pred ich dokončením.

Kapitola 4

Implementácia

Táto kapitola obsahuje popis implementácie môjho problému. Najprv popisujem výber techník detekcie, s ktorými budem pracovať a následne popisujem samotnú implementáciu uzlov. Na záver spomeniem priebeh a výsledky testovania na ARTable a robotovi PR2 v školskom robotickom laboratóriu.

4.1 Výber techník detekcie

Všetky techniky detekcie, ktoré som skúmal, boli teoreticky popísané v predchádzajcej sekcii 3.2. Pred tým, než som začal skúmať jednotlivé techniky detekcie objektov, bolo nutné zoznámiť sa s ROSom. Nainštaloval som ho na operačný systém Ubuntu 14.04. pomocou tutoriálov a rovnako pomocou návodov som prešiel dostupné cvičenia pre prácu s ROSom na wiki stránkach ROSu[1]. Pri technikách budem sledovať štyri aspekty spomenuté v sekcii 3.4.

Tieto kritéria alebo aspekty budem ďalej nazývať podľa poradia zoznamu, aj ako prvý až štvrtý aspekt/kritérium. Kde prvý, a to najdôležitejší aspekt je *schopnosť správne určiť o aký objekt sa jedná* a posledný štvrtý aspekt označuje *schopnosť rozoznať objekt skôr na základe tvaru ako na základe textúry, čo ho pokrýva.*

Balíček find_object_2d

Ako prvú metódu pre rozoznávanie a detekciu objektov som vyskúšal find_object_2d. Táto metóda je dostupná ako balíček pod ROSom. Po spustení sa zobrazí okno, pomocou ktorého je možné pridávať alebo mazať objekty z databázy, ktoré chceme porovnávať. Objekt je možné pridať priamo z aktuálneho obrazu, a to výrezom konkrétnej oblasti alebo ako akýkoľvek už dopredu vytvorený obrázok. Tieto obrázky sa porovnávajú s aktuálnym obrazom a v prípade zhody sa v obraze označí nájdený objekt.

Pomocou využitia Kinectov vie find_object_2d dodať požadované súradnice nájdených objektov, a tým pádom spĺňa druhé kritérium. Obmedzenia, ktoré vyplývajú z užívania tejto techniky sú ale veľké. Na jednej strane vie rozoznať aj objekt, ktorý je čiastočne prekrytý (obrázok 4.1), no na strane druhej, ak chceme, aby sme objekt mohli detegovať z každého uhla, je nutné pracne do databázy pridávať každý obrázok objektu z požadovaného uhla osobitne. Následne vzniká problém pri určení o aký objekt sa jedná, keďže každý obraz je považovaný za osobitný objekt a neexistuje žiaden príznak pre rozoznanie o aký objekt

[1] http://wiki.ros.org/ROS/Tutorials

sa vlastne jedná. Tým pádom nie je dostatočne splnené prvé kritérium, ktoré je kľúčové. Naviac pri netextúrovaných objektoch vzniká ďalší problém, kedy techniky ako SURF, ORB a iné nie sú schopné dostatočne rozoznávať objekty.

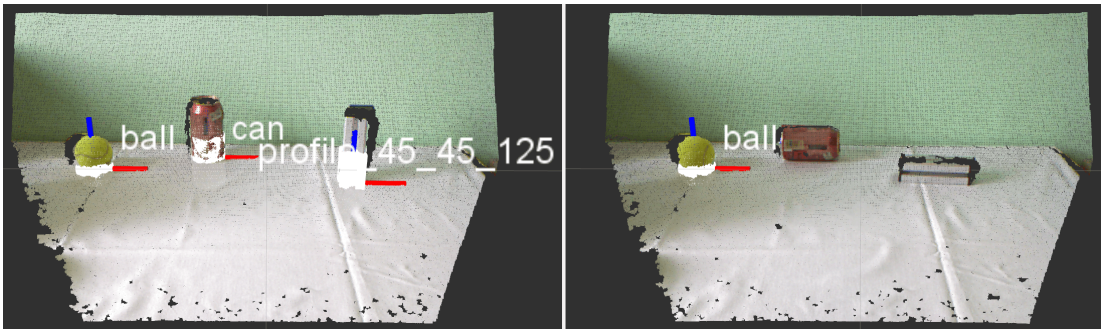
Balíček find_object_2d je teda pre môj problém nevhodný a nebudem ho v tejto práci ďalej využívať.

Obr. 4.1: Vľavo ukážka detekcie prednej strany krabice pomocou find_object_2d (vyznačené červeným orámovaním objektu). Vpravo ukážka detekcie rovnakého objektu pri čiastočnom prekrytí iným objektom.

Tabletop

Po nainštalovaní ORK je Tabletop prvou technikou, ktorú som v rámci ORK vyskúšal. Pred detekciou je najprv nutné nahrať do databázy všetky požadované objekty, ktoré chceme detegovať. ORK využíva databázu CouchDB, kde je nutné nahrať 3D modely objektov. Tieto modely akýchkoľvek objektov je možné stiahnuť z internetových stránok. Je treba si ale dávať pozor na veľkosť samotného modelu. ORK vyžaduje reálne veľkosti modelov podľa svojich skutočných predlôh. Preto je niekedy nutné 3D model objektu upraviť na požadovanú veľkosť, inak môže dôjsť k celkovému zlyhaniu detekcie.

Tabletop nepotrebuje žiadne natrénovanie objektov pred spustením samotnej detekcie. Pri správne vložených objektoch do databázy vie Tabletop presne určiť o aký objekt sa jedná. Pri tvarom podobných objektoch je pri nižšom rozlíšení hĺbkovej kamery možné aj chybné rozoznanie, o aký objekt sa jedná. Prvý aspekt je aj napriek možnej chybovosti dostatočne splnený a rovnako aj druhý aspekt. Pozíciu objektu vie určiť Tabletop presne aj pri detekcii viacerých objektov naraz. Hlavným obmedzením je ale nemožnosť akokoľvek položiť objekt na stôl. Na obrázku 4.2 je názorná ukážka, kde Tabletop nevie detegovať objekty položené inak ako vzpriamene postavené na stole. Všetky ORK techniky rozoznávajú tvary. Preto aj Tabletop spĺňa štvrtý aspekt o preferencii techniky, ktorá sa skôr zaoberá tvarom ako textúrou.



Obr. 4.2: Ukážka detekcie troch objektov (tenisová loptička, plechovka, hliníkový profil) pomocou Tabletop. Vľavo sú všetky objekty postavené vzpriamene (vertikálne) k stolu (detekcia každého objektu prebehla úspešne). Vpravo je plechovka a hliníkový profil položený horizontálne voči stolu. Tieto objekty tak už nie sú naďalej rozoznané.

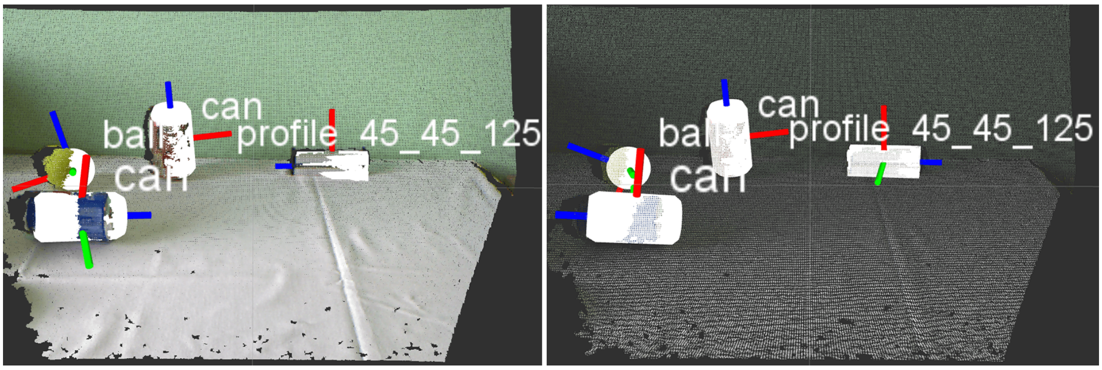
Tabletop rozhodne vyhovuje viac ako find_object_2d, no pre svoje obmedzenia nie je až tak efektívny ako Linemod, ktorý je popísaný v nasledujúcej sekcii. Preto ani Tabletop nebudem využívať pri implementácii môjho uzla.

Linemod

Druhou technikou ORK, ktorú som vyskúšal je Linemod. Táto technika rovnako vyžaduje pre detekciu 3D modely objektov v CouchDB databáze ORK. Pred samotnou detekciou je ale nutné spustiť natrénovanie objektov v databáze. Pri natrénovaní sa predvolene využíva OSMesa pre vykresľovanie (anglicky rendering) objektov pri rôznych uhloch. V dobe písania práce však OSMesa prestala podporovať vykresľovanie v ORK. Preto bolo nutné nastaviť alternatívne vykresľovanie pomocou GLUT renderer, ktorý vie plniť rovnakú funkciu ako OSMesa.

Prvý sledovaný aspekt Linemod spĺňa podobne ako Tabletop bez väčších problémov iba s miernými výpadkami detekcie, poprípade pri podobných tvaroch objektov môže nastať nie vždy správne rozoznanie o aký objekt sa jedná. Druhé kritérium o určení pozície spĺňa, kedy Linemod je schopný určiť pozíciu v priestore s takou presnosťou akou mu to dovolí snímacie zariadenie. Linemod výrazné obmedzenia detekcie nemá. Jediným obmedzením je nemožnosť detekcie čiastočne prekrytých objektov. To znamená, že je schopný detegovať objekty položené akokoľvek na stole aj mimo neho, o čom svedčí aj obrázok 4.3. Keďže Linemod rozoznáva tvar a textúru neberie do úvahy, aj štvrtý aspekt je mu naklonený. Pri detegovaní veľkého počtu objektov nastáva spomalenie a interval, kedy dochádza k detekciám sa predlžuje pre veľký objem dát, ktorý táto metóda spracúva.

Linemod je najlepším adeptom pre využitie pri vytváraní uzla pre ARTable a robota PR2. Keďže takmer nemá obmedzenia, Linemod dokáže detegovať objekty kdekoľvek, aj bez nutnosti stola. Svojou veľkou flexibilitou sa Linemod technika stáva najlepšou ORK technikou, ktorá ma najbližšie k práci v reálnom svete medzi ľudmi. Využijem ju preto pri tvorbe môjho uzla.

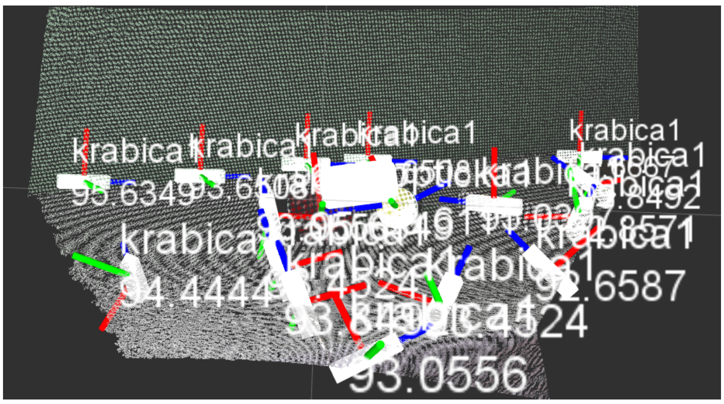


Obr. 4.3: Ukážka detekcie viacerých objektov pomocou techniky Linemod. Z obrázkov je možné všimnúť si, že nezáleží v akej póze je objekt pred kamerou.

Texture Object Detection

Keďže prácu s priehľadnými objektmi na ARTable neuvažujeme, nebolo potrebné skúmať techniku Transparent Objects. Tým pádom poslednou technikou, ktorú som v rámci ORK vyskúšal je TOD (Texture Object Detection). Táto technika pracuje iba s objektmi, ktorých 3D model bol vytvorený manuálne. Okrem tvaru objektu berie do úvahy aj textúru objektu. Pred spustením trénovania a detekcie objektov je nutné nastaviť algoritmy pre detekciu a popis príznakov. Mojou voľbou oboch algoritmov bol ORB.

Schopnosť určiť o aký objekt sa jedná TOD spĺňa iba z časti. Je tomu tak preto, že TOD, ako som spomínal, pracuje výlučne iba s manuálne vytvorenými 3D modelmi (nepodporuje modely vytvorené akýmkoľvek softwareom). Pomocou uzla `object_recognition_capture` je možné tieto 3D modely objektov vytvoriť. Pomocou otáčacej platformy a špeciálneho vzoru, v ktorom strede sa nachádza požadovaný objekt, vie tento uzol poskladať obraz z viacerých uhlov objektu a zrekonštruovať tak 3D model objektu aj s textúrou. Problémom je, že tieto modely nie sú tak dokonalé ako modely vytvorené v špecializovaných programoch. Výsledkom je, že TOD často pri detekcii nevie rozoznať o aký objekt sa jedná alebo naopak, ak je textúra podobná ako okolité prostredie, deteguje veľa falošne pozitívnych objektov (obrázok 4.4).



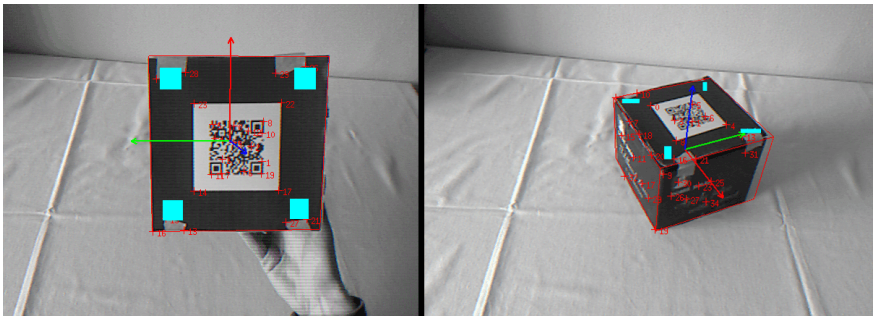
Obr. 4.4: Ukážka detekcie množstva falošne pozitívnych objektov pomocou metódy TOD.

TOD teda nevyhovuje mojej najdôležitejšej požiadavke, a preto s ním nebudem ďalej pracovať. Rovnako vytváranie vlastných 3D modelov je niekoľkonásobne zdĺhavejšie ako jednoduché stiahnutie objektu z webu a poprípade jeho upravenie.

ViSP

ViSP je v ROSe dostupný ako zoskupenie viacerých balíčkov. Zameral som sa na balíček `visp_auto_tracker`, keďže nevyžaduje zakaždým manuálnu inicializáciu objektu klikaním na vopred určené body objektu. O inicializáciu sledovania objektu sa stará QR kód umiestnený na objekte. Pred detekciou a sledovaním objektu je najprv nutné v konfiguračných súboroch a súboroch modelu zadať presné súradnice bodov predstavujúcich rohy QR kódu umiestneného na objekte a súradnice samotných vrcholov objektu. Pomocou tohto presného modelu tak ViSP vie na základe QR kódu inicializovať objekt a sledovať jeho pohyb ďalej v priestore.

Prvý sledovaný aspekt ViSP spĺňa, keďže je možné pre každý objekt zvoliť unikátny QR kód. Tým je možné presne určiť o aký objekt sa jedná. Rovnako vie určiť presnú pozíciu objektu. Obmedzením je nutnosť viditeľnosti QR kódu na objekte. Pokiaľ je QR kód viditeľný, tak je možná akákoľvek manipulácia s objektom. Po strate QR kódu sa uloží posledná viditeľná pozícia a hľadá sa QR kód pre obnovenie sledovania. Keďže ViSP sleduje objekt na základe hrán a vrcholov, rozoznáva ho teda na základe tvaru, a preto je splnený aj štvrtý mnou sledovaný aspekt.



Obr. 4.5: Ukážka detekcie objektu pomocou techniky ViSP. Vľavo inicializácia detekcie a sledovania objektu, vpravo sledovanie objektu.

ViSP dostatočne spĺňa všetky moje kritéria, a preto ho využijem pri tvorbe môjho uzla pre využitie na ARTable.

Zhodnotenie výberu techník

Pre ďalšiu prácu a implementáciu uzlov som si vybral ORK techniku Linemod a ViSP. Vzniknú teda dva osobitné a na sebe nezávislé balíčky, ktoré budú obsahovať jednotlivé uzly. Pri následnom bližšom skúmaní, testovaní týchto dvoch techník, som došiel k záveru, že Linemod je vhodný pre menšie až stredne veľké objekty. Medzi ne patria napr. objekty tvarom gúľ, valcov, kvádrov ale aj rôzne hranoly a iné objekty, ktoré majú ťažšie popísateľný tvar ako žiarovka alebo model lietadla či auta. Objekty vhodné pre detekciu pomocou ViSPu by mali mať ideálne stredne veľkú až veľkú veľkosť kvôli potrebnému QR kódu, aby bol dostatočne čitateľný. Tvarom to môžu byť rôzne druhy krabíc a objektov s jasne danými

hranami a vrcholmi. ViSP si vie poradiť s komplexnými, zložitými a veľkými objektmi ako napr. model hradu či celé časti nábytku a pod.

4.2 Implementácia ORK uzla

Tento uzol má slúžiť ako most medzi ORK a ARTable s PR2. ROS správy prúdiace z ORK vytvárané pomocou metódy Linemod nie sú kompatibilné s ARTable. Je teda nutné tieto správy upraviť na požadovaný tvar a pridať všetky potrebné dáta pre správnu prácu ARTable. Uzol je implementovaný v jazyku C++.

Výstup ORK a CouchDB

Linemod pri detekcii objektov posiela na výstup dáta opakovane v časových intervaloch. Časový interval zasielania týchto správ sa pri zvyšovaní počtu detegovaných objektov zvyšuje. Pri jednom objekte to môže byť aj niekoľkokrát za sekundu, kým pri piatich objektoch to môžu byť už jednotky sekúnd. Výstup tvorí jedna ROS správa vo forme poľa, ktorá obsahuje informácie o všetkých detegovaných objektoch, ktoré boli v danom čase rozoznané. Každý prvok poľa predstavuje jeden rozpoznaný objekt. Správa obsahuje okrem štandardných ROS hlavičiek množstvo informácii o konkrétnom objekte. Pre mňa dôležitá je informácia o type objektu (informácia o aký objekt sa jedná, napr. plechovka). Správa obsahuje aj rovnako potrebné informácie o pozícii a orientácii objektu v priestore.

Databázou ORK je CouchDB. Pri pridávaní objektu do nej, vyžaduje okrem cesty k súboru s 3D modelom, zadať aj typ objektu a krátky popis k objektu. Typ objektu a unikátne ID, ktoré je generované pri pridaní objektu do databáze, môžeme považovať za rovnako unikátne hodnoty, ktoré odkazujú na seba. To ale platí iba v prípade, že v databáze nebudú pridané objekty s rovnakým typom viacnásobne (viaceré unikátne ID by tak odkazovali na rovnake typy objektov).

Vstup ARTable a AR databáza

Pre správnu funkčnosť potrebuje ARTable na svojom vstupe o každom objekte nasledujúce informácie. Unikátne meno/ID každého objektu, typ objektu o aký objekt sa jedná a pozíciu v priestore (pozícia a orientácia). Tieto dáta očakáva vo forme poľa, kde každý prvok poľa predstavuje jeden konkrétny objekt. Dáta sa následne spracúvajú pre ďalšiu prácu ARTable, napríklad aj pre robota PR2.

Každý objekt uložený v AR databáze, obsahuje informáciu o type objektu a rozmeroch (šírka, hĺbka, výška objektu). Typ objektu v AR databáze predstavuje ekvivalent k typu objektu v CouchDB databáze. Preto je potrebné, aby tieto dva atribúty boli naprieč rôznym databázam rovnaké a v jednej aj druhej databáze sa nevyskytovali viaceré objekty s rovnakým typom.

Problémom je, že ORK posiela iba informáciu o type objektu. To znamená, že ak sa detegujú tri rovnaké objekty, ORK pošle správu o nájdení troch rovnakých objektoch, s rovnakým ID, ktoré sa líšia iba v pozícii. Po ďalšom časovom intervale pošle rovnakú správu, kde sa ale nezaručuje, že poradie nájdených objektov je rovnaké ako v predchádzajúcej správe. Každá detekcia je teda nezávislá na predchádzajúcich detekciách a nemá s nimi žiaden súvis.

Znamená to, že aby môj uzol zaručil unikátne ID každéu objektu na stole, musí plniť aj funkciu akéhosi sledovania objektov. Len tak bude zaručený správny vstup pre ARTable,

keďže ARTable vyžaduje unikátne meno pre každý objekt. Pokiaľ sa objekt zo stola neodoberie, mal by byť stále pokladaný za rovnaký, aj pri manipulácii s ním ako napríklad pri jeho posúvaní po stole.

Príprava potrebných dát

Na začiatku prebehne inicializácia uzla. Nastaví sa odber ORK správ pre ďalšie spracovanie a inicializuje sa objekt pre posielanie správ pre ARTable. Rovnako sa testuje, či bežia potrebné ROS služby a v kladnom prípade sa vytvoria potrební klienti pre neskoršie volanie týchto služieb.

Po obdržaní správy z ORK sa táto správa začne spracovávať. Ako som už spomínal, táto správa ma formu poľa. Ku každému prvku poľa čiže objektu sa dostanem pomocou cyklu, ktorý prejde všetky detegované objekty. Následne sa zavolá služba pre získanie typu objektu z CouchDB pomocou predania ID objektu. V ORK správe je totiž uložené iba ID. Aby som mohol pracovať aj s AR databázou, potrebujem typ objektu, na ktorý odkazuje konkrétne ID. Získaný typ objektu využijem pri volaní služby pre zistenie rozmerov objektu, ktoré sú uložené v AR databáze.

Keďže mám všetky potrebné informácie o detegovanom objekte, uzol začne vykonávať funkciu sledovania objektov. No najprv v prípade, že sa jedná o vôbec prvú detekciu po spustení uzla, uložím prvý objekt priamo do prázdneho vektora, ktorého každý prvok tvorí štruktúra s informáciami o objekte. Prvky tohto vektora predstavujú aktuálne objekty položené na stole. Štruktúra obsahuje unikátne ID objektu, ako ho vyžaduje ARTable. Zavolaná funkcia vygeneruje toto ID, ktoré sa skladá z mena typu objektu a čísla, ktoré je každým novým objektom vloženým do vektora inkrementované. Tým je zaručené, že každý objekt na stole má osobitné ID. Ďalej štruktúra obsahuje názov typu objektu, ktorý preberá z databáz a pozíciu spoločne s orientáciou, ktorú čerpá z ORK správy. Tieto dáta sú pripravené pre odoslanie pre ARTable. Naviac obsahuje štruktúra interné informácie pre prácu uzla. Ide o príznak, ktorý slúži pri porovnávaní objektov a časové razítko pre využitie starnutia objektov.

Hlavná časť uzla

O funkciu sledovania (anglicky tracking) objektov sa stará hlavná časť uzla. Každý detegovaný objekt z ORK správy je samostatne po jednom porovnávaný so všetkými objektmi v mojom vektore objektov pomocou cyklu, ktorý prejde celý vektor. Najprv sa zistí, či sa jedná o rovnaký typ objektu. V kladnom prípade sa začne porovnávať pozícia. Pre každú os sa volá rovnaká funkcia, ktorá na vstupe potrebuje súradnicu objektu na danej osi, rovnako súradnicu porovnávaného objektu a šírku alebo výšku alebo hĺbku objektu (podľa toho o akú os sa jedná), ktorú som získal z AR databáze pomocou spomínanej služby. V prípade, že sa objekt aspoň z malej časti prekrýva (postačuje dotyk) na každej osi s porovnávaným objektom z vektora, ide o ten istý ale posunutý objekt. Následne sa aktualizuje pozícia objektu vo vektore a nastaví sa príznak, že bol objekt aktualizovaný. Taktiež sa aktualizuje aj časové razítko.

V prípade, že sa porovnávané objekty neprekrývajú, nastaví sa príznak o aktualizácii objektu negatívne. Rovnako sa príznak nastaví, ak porovnávané objekty nie sú rovnakého typu. Po postupnom porovnaní objektu poslaného z ORK so všetkými objektmi uloženými vo vektore, sa na záver skontrolujú všetky príznaky, či bol nejaký objekt aktualizovaný. Ak žiaden objekt nebol aktualizovaný (značí to, že nebola nájdená žiadna zhoda), vloží sa úplne nový objekt s novým ID do vektora.

Pri každom vytvorení nového objektu alebo aktualizovaní sa prepíše časová hodnota na aktuálny čas v premennej objektu, v ktorej je uložený unixový čas. Tento čas sa pred samotným vytvorením správy pre ARTable u každého objektu skontroluje a v prípade, že bol objekt neaktualizovaný dlhšie ako určitá časová konštanta vypočítaná zo správania sa na konkrétnom počítači, je odstránený z aktuálnych objektov na stole. Tým je dosiahnuté starnutie objektov. Pre udržanie objektu na stole musí byť aktualizovaný aspoň raz za daný časový interval. Tým sa odstraňuje aj problém náhodného výpadku detekcie, kedy ORK nie je schopný vždy správne detegovať všetky objekty na stole a problém dlhšieho časového intervalu medzi detekciami, ako ukázalo testovanie 4.4.

Po porovnaní všetkých objektov a skontrolovaní aktuálnosti objektov na stole je vygenerovaná ROS správa pre ARTable. Cyklom sa vytvorí pole objektov s ich potrebnými atribútmi pre ARTable a doplní sa o hlavičku. Po publikovaní správy uzol čaká na správu z ORK o detegovaných objektoch a celý mechanizmus sa spúšťa odznova. Takto sa cyklí až pokiaľ používateľ tento uzol nevypne.

Uzol pri sledovaní neporovnáva presnú pózu objektu. Teda neberie do úvahy orientáciu objektov a porovnáva ich iba so základným postavením, ako boli nahrané do databázy. Keďže sledovanie objektu má na ARTable primárne slúžiť na to, aby bola zaručená unikátnosť objektu na stole a nepredpokladá sa práca na ARTable s výrazne pohybujúcimi sa objektmi pri rôznych pózach na stole, ale predpokladá sa práca s objektmi najčastejšie vo vzpriamenej polohe bez ich výrazného pohybu, je toto riešenie postačujúce.

4.3 Implementácia ViSP uzla

Keďže samotná metóda ViSP sa stará o sledovanie objektu, tento uzol má za úlohu iba upravovať prijaté správy z ViSPu a preposielať ich pre ARTable. Aj tento uzol bol implementovaný pomocou jazyka C++.

Výstup a spúšťanie ViSPu

Technika ViSP publikuje v ROSe niekoľko druhov správ. Mňa zaujímajú správy s informáciou o pozícii objektu a správa s ID detegovaného objektu. Samotná technika ViSP neumožňuje detekciu viacerých objektov naraz. Pri spustení uzlov ViSPu je potrebné zadať pomocou parametrov aký typ objektu sa bude detegovať a aký odkaz je zakódovaný v QR kóde, ktorý slúži na inicializáciu, čo je vlastne ID objektu. Mnohonásobným spustením tohto uzla, pričom každý uzol deteguje iný objekt, je možné dosiahnuť detekciu viacerých objektov naraz. To znamená, že môj uzol sa bude spúšťať rovnako veľa krát s každým spustením ViSPu podľa počtu objektov. Týmto spôsobom ViSP začne posielať množstvo správ.

Pre kompatibilitu ARTable s ViSPom je nutné dodržať rovnaké názvy typov objektov v AR databáze a pri zadávaní parametra odkazujúceho na typ objektu pred spustením ViSPu.

Hlavná časť uzla

Každé spustenie uzla sa stará iba o jediný objekt. Po inicializácii uzol začne odoberať správy s informáciou o odkaze QR kódu a o pozícii a orientácii objektu. Najprv sa spracuje správa z QR kódu. QR kód obsahuje odkaz s unikátnym názvom/ID objektu. Textový reťazec ID sa skladá z názvu typu objektu a čísla. Pomocou funkcie sa tento textový reťazec spracuje,

čím sa zistí typ objektu, aký sa práve deteguje. Preto je potrebné dodržiavať spomínaný tvar ID reťazca.

Následne po obdržaní ROS správy aj o pozícii objektu uzol vygeneruje celú správu pre ARTable. Doplní ju ešte o ROS hlavičku a následne ju publikuje. Takýmto spôsobom uzol publikuje správy, kým nie je vypnutý alebo prestane dostávať správy o objekte.

Keďže ViSP posiela údaje o pozícii objektu neustále, aj keď ešte nie je detegovaný (súradnice sú v tomto prípade rovné nule) a aj po strate objektu z obrazu (hodnoty súradníc sú rovné poslednému známemu miestu, kde sa objekt nachádzal), bolo nutné tieto stavy ošetriť. Pomocou podmienok sa zisťuje, či sú súradnice nulové alebo či sa presne zhodujú s poslednou detekciou. V kladnom prípade sa považuje objekt za neprítomný na stole a nepublikuje sa žiadna správa pre ARTable.

4.4 Testovanie

Testovanie mojich uzlov prebiehalo postupne počas vývoja. Po každej implementovanej časti sa hneď overovala funkčnosť. Najprv sa funkčnosť uzlov overovala pomocou Kinect V1 kamery. Keďže sa v robotickom laboratóriu využívajú aj kamery Kinect V2, chcel som vyskúšať prácu uzlov a samotných techník detekcií aj na tejto kamere. Ukázalo sa, že technika Linemod bola vyvíjaná primárne pre Kinect V1 a jej využitie na kamere Kinect V2 bolo výrazne obmedzené a aj po úpravách parametrov a samotného zdrojového kódu Linemod techniky, vykazovala spustená detekcia oveľa horšie výsledky ako detekcia pomocou Kinect V1. Technika ViSP pri pokusoch s Kinect V2 úplne zlyhávala. Preto testovanie prebiehalo výlučne pomocou Kinect V1.

Keďže správna funkčnosť mojich uzlov je závislá na presnosti a správnosti metódy detekcie objektov, rozhodol som sa testovať a zhodnotiť presnosť metód detekcie. Po konzultácii s vedúcim práce som sa zameral na techniku Linemod, keďže v tejto chvíli sa viac hodí využitie Linemod v robotickom laboratóriu.

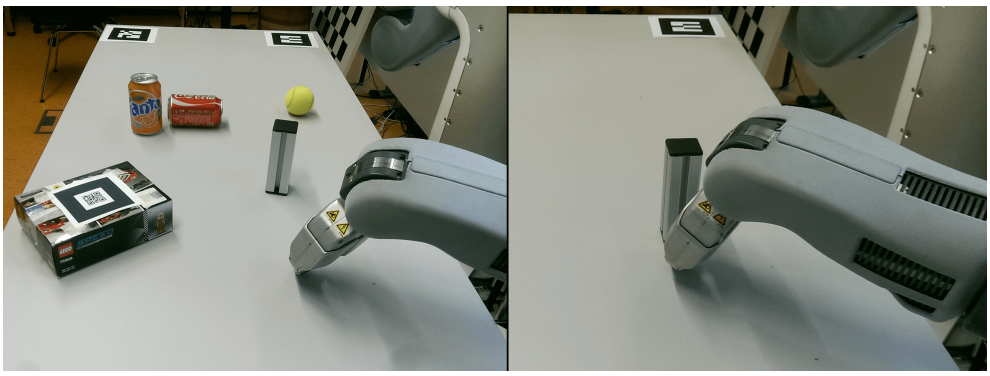
Pri testovaní budem sledovať správnosť klasifikovaného, detegovaného objektu ako aj presnosť určenia správnej pozície detegovaného objektu. Budem teda sledovať kombináciu týchto dvoch aspektov.

Testovacia dátová sada

Pre testovanie bolo vytvorených päť dátových sád v podobe bag súborov. Každý bag súbor tvoril jednu minútu dlhý záznam z Kinectu V1, ktorý obsahoval potrebné dáta pre prácu Linemod. Išlo o hĺbkové dáta aj o súradnice skutočných pozícií objektov umiestnených v obraze.

V databáze ORK a teda natrénované objekty sú štyri. Ide o hliníkový profil, tenisovú loptičku, plechovku a krabicu. Keďže sa v laboratóriu pracuje najčastejšie s hliníkovým profilom, prvý bag súbor obsahoval záznam, na ktorom je jeden hliníkový profil položený na stole. Druhý súbor obsahoval dva profily položené na stole. Pre vyskúšanie detegovania ďalších objektov, tretí bag súbor obsahoval dve plechovky a jednu tenisovú loptičku. Najväčšiu zátaž predstavoval bag súbor, ktorý obsahoval päť objektov, a to dve odlišne položené plechovky, loptičku, hliníkový profil a krabicu. Taktiež bola na stôl schválne položená ruka robota, ktorú by mal Linemod ignorovať, keďže v databáze sa takýto objekt nenachádza (scéna zachytená na obrázku 4.6).

Pre vyskúšanie úspešnosti detekcie čiastočne prekrytých objektov posledný bag súbor obsahuje hliníkový profil, ktorý je ale čiastočne prekrytý rukou robota ako vidieť na obrázku 4.6.



Obr. 4.6: Vľavo ukážka scény s piatimi objektmi, vpravo ukážka prekrytého hliníkového profilu.

Sledované aspekty

Pre hodnotenie testov (pre vyhodnotenie správnosti určovania objektov a presnosti pozície detegovaných objektov) bude využitá ROC krivka, ktorá ilustruje výkonnosť Linemod pri jednotlivých testoch, testovaných na spomínaných bag súboroch. Keďže ROC krivka vyjadruje vzťah medzi senzitivitou a špecificitou, je potrebné pri testovaní sledovať skutočne pozitívne (True Positive, TP) prípady detekcie, skutočne negatívne (True Negative, TN), falošne pozitívne (False Positive, FP) a falošne negatívne (False Negative, FN) prípady detekcie. Následne sa z počtov týchto prípadov vypočítajú hodnoty senzitivity a špecificity a vykreslí sa krivka v grafe.

Priebeh testov

Pre sledovanie jednotlivých prípadov detekcie bol vytvorený testovací program, ktorý bol založený na mojom ORK uzle. Hlavné telo programu bolo upravené tak, aby pri každej detekcii rátalo výskyt TP, FP, TN a FN prípadov za plnej kontroly typov jednotlivých detegovaných objektov. Skutočné pozície objektov, ktoré sa majú sledovať, boli na základe vložených značiek v bag súboroch vložené priamo do testovacieho programu. Tieto pozície sa porovnávali s presnosťou na 1,5 centimetra. V prípade, že bol objekt detegovaný v rozmedzí pre osi X, Y a Z do 1,5 centimetra vrátane, táto detekcia sa považovala za skutočne pozitívnu.

Testovalo sa na zostave so špecifikáciou procesoru Intel Core i7 4720HQ a 8GB DDR3L pamäte RAM. Každý bag súbor bol pustený v slučke po dobu dvoch hodín a po ukončení testu nasledovalo vyhodnotenie konkrétneho testu. Je treba podotknúť, že ak sa detegujú viaceré objekty na stole, jedna publikovaná správa detekcie môže obsahovať informáciu o viacerých detegovaných objektoch naraz. Počet celkových detekcií sa nemusí rovnať celkovému počtu detegovaných objektov.

Vyhodnotenie testov

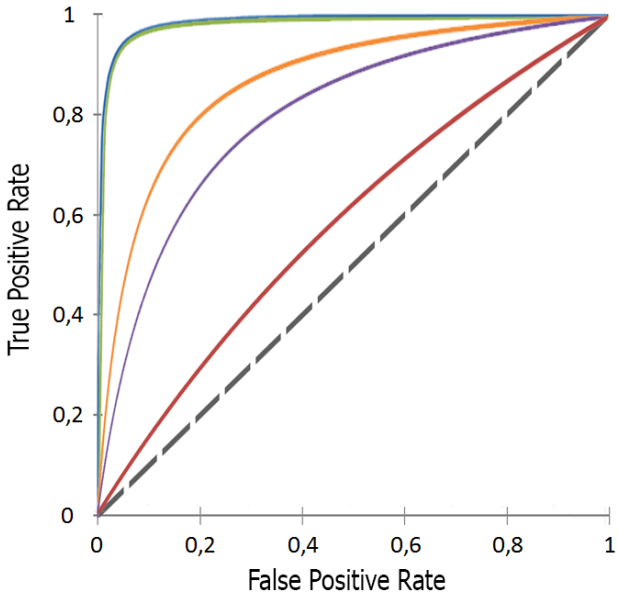
Ako prvý bol vyhodnotený bag súbor s jedným profilom na stole. Linemod vyhodnotil za 2 hodiny vyše 26 000 detekcií, čo činí 3,7 detekcií za sekundu. Senzitivita alebo aj inak miera skutočne pozitívnych testov bola 93,6%. Detekcie mali aj vysokú prediktívnu hodnotou pozitívneho testu a to až 95,5%.

Druhý test s dvoma profilmi na stole ukázal, že pre väčšie spracovanie dát, Linemod spracoval a publikoval o 10 000 detekcií menej za rovnaký časový interval (celkovo 15 000 detekcií). Znamená to kadenciu 2 detekcie za sekundu. Prázdnych detekcií bolo oproti prvému testu posielaných menej. No naopak pribudlo viac falošne pozitívnych detekcií. Celkový výsledok druhého testu je ale veľmi podobný ako výsledok prvého, o čom svedčia aj krivky v ROC grafe, ktoré sa takmer prekrývajú (obr. 4.7).

Trend klesania počtu publikovaných správ Linemodu pre väčšie množstvo objektov na stole pokračoval aj pri treťom teste pri troch objektoch na stole, kde publikoval necelých 2 000 detekcií. Toto spomalenie predstavuje 2,6 detekcií za 10 sekúnd. Senzitivita testu bola 79%.

Pri teste s veľkou záťažou s piatimi objektmi Linemod vyprodukoval okolo 650 správ detekcií, čo činí pokles počtu detekcií o 97,5% oproti prvému testu. Za 10 sekúnd to predstavuje 0,9 detekcie. Miera skutočne pozitívnych detekcií klesla oproti predchádzajúcim testom na približne 72%.

Posledným testom bol test pri čiastočnom prekrytí objektu. Linemod obmedzením ako som opísal v 4.1 je, že nevie detegovať čiastočne prekryté objekty. Testy ukázali senzitivitu približne 7%. Väčšina detekcií bolo prázdnych, kde Linemod nezistil žiaden objekt na scéne. Prediktívna hodnota pozitívneho testu sa pohybovala okolo 62%, čo ukazuje, že ak bol nejaký objekt detegovaný, tak v 62% prípadoch bol správne určený.



Obr. 4.7: ROC krivky pre jednotlivé testy. Jeden objekt na stole (modrá), dva objekty na stole (zelená), tri objekty (oranžová), päť objektov (fialová), jeden prekrytý objekt (červená).

Záver testov

Z testov sa dá odvodiť, že Linemod pri zvýšení počtu detegovaných a spracovávaných objektov na stole znižuje počet detekcií za rovnakú jednotku času. Je to spôsobené tým, že Linemod porovnáva potenciálne objekty s tisíckami pohľadov na objekty vytvorenými pri natrénovaní objektov. S viacerými objektmi na stole tak Linemod musí pracovať s väčším objemom dát, a to spôsobuje celkové spomalenie detekcií. Rovnako pri väčšom počte objektov je tak možné chybne určiť viaceré objekty, a tým stúpa aj celková chybovosť rozoznaných objektov. Kombináciou týchto aspektov tak pri väčšom počte objektov na stole klesá celková výkonnosť techniky Linemod, čo aj dokazuje spomínaný graf 4.7.

Pri poslednom teste sme sa presvedčili, že úspešnosť detekcie čiastočne prekrytých objektov je malá, no nie nulová. Z týchto testov bola v mojom ORK uzle odvodená hodnota 12 sekúnd, ktorá predstavuje časový interval starnutia (popísané v implementácii uzla 4.2).

Overenie funkčnosti uzlov

Pre overenie úplnej funkčnosti mojich uzlov boli uzly spúšťané a testované v robotickom laboratóriu na ARTable. Spustený bol aj ARTable uzol pre spracovanie dát, pre ktorý moje uzly publikujú. Týmto bola úspešne overená správnosť, funkčnosť a kompatibilita finálnych verzií mojich uzlov s ARTable.

Robotovi PR2 je tak umožnená práca s objektmi na stole bez AR štítkov. Objekty môže presúvať, otáčať a ľubovoľne s nimi manipulovať.

Kapitola 5

Záver

Cieľom tejto práce bolo vytvoriť modul pre robota PR2 v školskom robotickom laboratóriu, pre detekciu objektov umiestnených pred ním a umožnenie práce s nimi.

Po naštudovaní teórie detekcie objektov v obraze a princípov práce s ROSom (Robotický Operačný Systém) a Kinectom som sa zameral na návrh a implementáciu modulu. Mnou vytvorený modul vo forme ROS uzlov ponúka využitie dvoch existujúcich techník detekcie objektov. Uzly slúžia na prepojenie týchto techník detekcie objektov so stolom ARTable, s ktorým pracuje PR2. Prvý uzol využíva ORK (Object Recognition Kitchen) techniku nazvanú Linemod, pri ktorej môj implementovaný uzol sleduje všetky detegované objekty a zaručuje správny vstup pre ARTable. Druhý implementovaný uzol pracuje s technikou ViSP. Detekcie boli získavané pomocou kamery Kinect od firmy Microsoft.

Zistilo sa, že využitie novšej, druhej verzie Kinect kamier je výrazne obmedzené a nepodarilo sa mi ich pri tejto práci využiť. Pracoval som preto primárne so staršou verziou Kinectu. Predmetom ďalšieho vývoja ORK uzla by mohla byť presnejšia implementácia porovnávania sledovaných objektov, kde v tejto chvíli zanedbávam konkrétnu orientáciu objektu a sledovanie (anglicky tracking) objektov je tak založené iba na porovnávaní základnej pózy modelu objektu. Keďže toto sledovanie má hlavne za úlohu zaručiť unikátnosť každého položeného objektu na stole pre správny vstup ARTable, je tento spôsob sledovania objektov v laboratóriu zatiaľ dostatočný. Funkčnosť uzlov bola riadne overená a testovaná priamo v školskom robotickom laboratóriu na ARTable a robotovi PR2.

Literatúra

[1] *Algorithm Details - Object Recognition Transparent Objects*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/object_recognition_core/

[2] *Documentation - Point Cloud Library (PCL)*. [Online; navštívené 07.02.2017].
URL http://pointclouds.org/documentation/tutorials/walkthrough.php

[3] *Feature Matching*. [Online; navštívené 19.02.2017].
URL http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[4] *Find_Object_2D*. [Online; navštívené 21.02.2017].
URL http://wiki.ros.org/find_object_2d

[5] *Hardware Specs*. [Online; navštívené 06.02.2017].
URL https://www.willowgarage.com/pages/pr2/specs

[6] *Object Recognition Kitchen*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/object_recognition_core/

[7] *Object Recognition Using LINE-MOD*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/linemod/index.html

[8] *ORB (Oriented FAST and Rotated BRIEF)*. [Online; navštívené 08.02.2017].
URL http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html

[9] *Random view Generator - object_recognition_renderer*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/ork_renderer/index.html

[10] *Tabletop Object Recognition*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/tabletop/index.html

[11] *Textured Object Detection*. [Online; navštívené 21.02.2017].
URL http://wg-perception.github.io/tod/index.html

[12] *ViSP - Open source visual servoing platform library*. [Online; navštívené 26.02.2017].
URL http://visp.inria.fr

[13] *ViSP - ROS*. [Online; navštívené 26.02.2017].
URL http://wiki.ros.org/vision_visp

[14] Bradski, G.; Kaehler, A.: *Learning OpenCV*. O'Reilly Media, 2008, ISBN 978-0-596-51613-0.

[15] Cousins, S.: ROS on the PR2 [ros topics]. *IEEE Robotics & Automation Magazine*, ročník 17, č. 3, 2010: s. 23–25.

[16] Jana, A.: *Kinect for Windows SDK programming guide: build motion-sensing applications with Microsoft's Kinect for Windows SDK quickly and easily*. Packt Publishing, 2012, ISBN 978-1-84969-238-0.

[17] Landa, J.; Procházka, D.: Usage of Microsoft Kinect for augmented prototyping speed-up. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, ročník 60, č. 2, 2013: s. 175–180.

[18] Mair, E.; Hager, G. D.; Burschka, D.; aj.: *Adaptive and Generic Corner Detection Based on the Accelerated Segment Test*. September 2010.

[19] Martinez, A.; Fernández, E.: *Learning ROS for robotics programming: a practical, instructive, and comprehensive guide to introduce yourself to ROS, the top-notch, leading robotics framework*. Packt Publishing, 2013, ISBN 978-1-78216-144-8.

[20] Quigley, M.; Conley, K.; Gerkey, B. P.; aj.: *ROS: an open-source Robot Operating System*. 2009.

[21] Richard, S.: *Computer Vision: Algorithms and Applications*. Springer London, 2010, ISBN 978-0-596-51613-0.

[22] Rusu, R. B.; Cousins, S.: 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[23] Valgma, L.: *3D Reconstruction Using Kinect v2 Camera*. Tartu Ulikool, 2016. URL https://dspace.ut.ee/handle/10062/51864

Prílohy

Príloha A

Obsah CD

Adresáre a súbory na priloženom CD:

- adresár `object_detection_ork/` - ROS balíček s implementovaným ORK uzlom, vrátane README súboru, ktorý obsahuje návod ako spustiť uzol,
- adresár `object_detection_visp/` - ROS balíček s implementovaným ViSP uzlom, vrátane README súboru, ktorý obsahuje návod ako spustiť uzol,
- adresár `tex/` - všetky zdrojové súbory potrebné pre LaTeX, pre vytvorenie tohto dokumentu,
- súbor `Detekce_objektu_na_stole.pdf` - tento dokument,
- súbor `video.mp4` - krátke video prezentujúce túto prácu.